STC12C5A60AD系列单片机器件手册
STC12C5201AD系列单片机器件手册

--- 1个时钟 / 机器周期8051
--- 无法解密
--- 低功耗,超低价
--- 高速，高可靠
--- 强抗静电，强抗干扰

STC12C5A08，12C5A08AD， 12C5A08S2
STC12C5A16，12C5A16AD， 12C5A16S2
STC12C5A20，12C5A20AD， 12C5A20S2
STC12C5A32，12C5A32AD， 12C5A32S2
STC12C5A40，12C5A40AD， 12C5A40S2
STC12C5A48，12C5A48AD， 12C5A48S2
STC12C5A52，12C5A52AD， 12C5A52S2
STC12C5A56，12C5A56AD， 12C5A56S2
STC12C5A60，12C5A60AD， 12C5A60S2
STC12C5A62，12C5A62AD， 12C5A62S2

宏晶科技
www.MCU-Memory.com
Update date: 2008-11-22

STC12C5A60AD 系列单片机器件手册
STC12C5201AD 系列单片机器件手册

--- 1 个时钟 / 机器周期 8051
--- 无法解密
--- 低功耗,超低价
--- 高速，高可靠
--- 强抗静电，强抗干扰

STC12C5201，12C5201PWM，12C5201AD
STC12C5202，12C5202PWM，12C5202AD
STC12C5204，12C5204PWM，12C5204AD
STC12C5205，12C5205PWM，12C5205AD
STC12C5206，12C5206PWM，12C5206AD
STC12LE5201,12LE5201PWM,12LE5201AD
STC12LE5202,12LE5202PWM,12LE5202AD
STC12LE5204,12LE5204PWM,12LE5204AD
STC12LE5205,12LE5205PWM,12LE5205AD
STC12LE5206,12LE5206PWM,12LE5206AD

宏晶科技
www.MCU-Memory.com
Update date: 2008-11-22

超强抗干扰 无法解密 宏晶新一代8051单片机

—— 8051 单片机全球第一品牌，中国 MCU 领航者

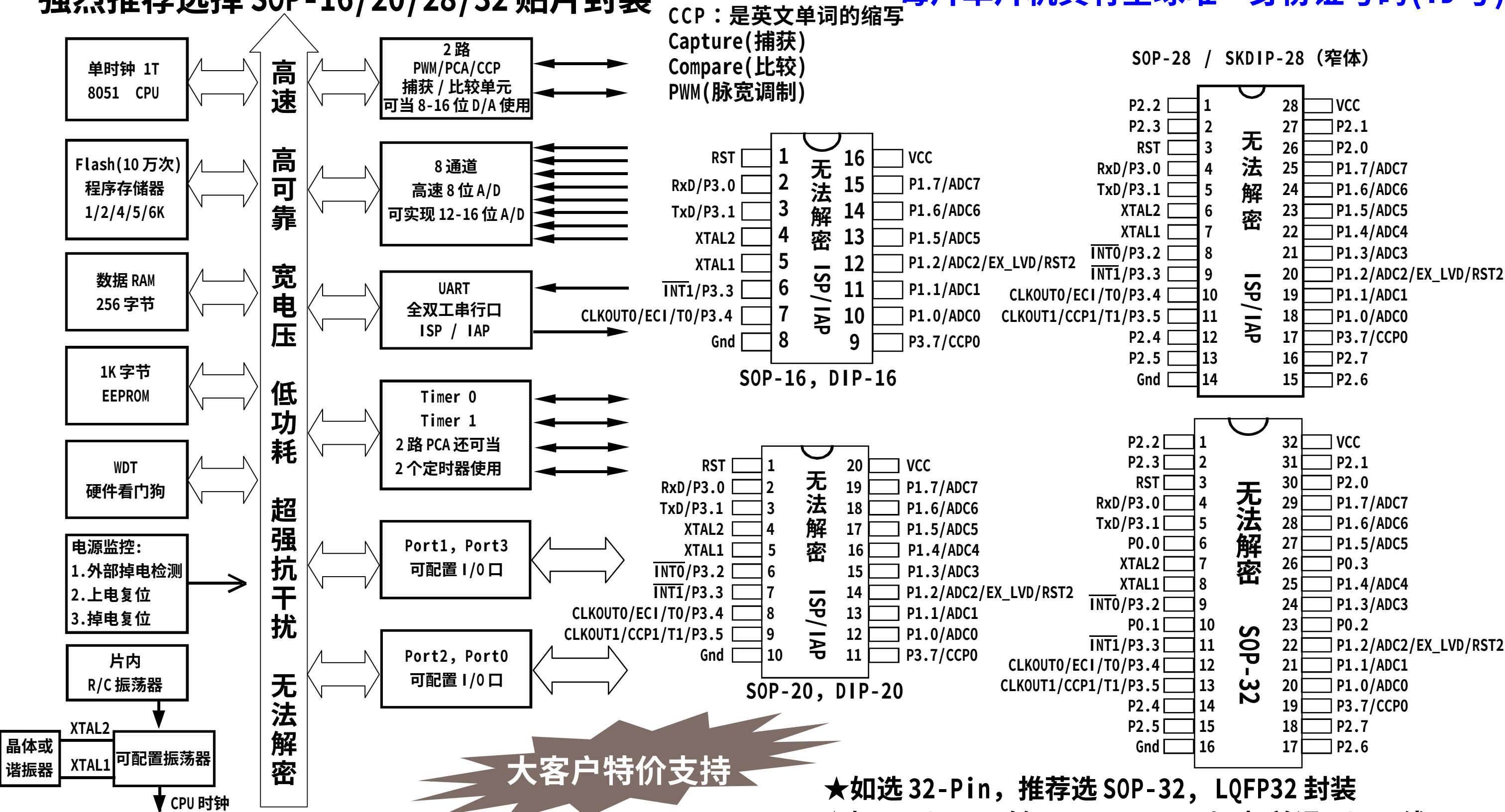
宏晶 STC12C5201AD 系列 1T 8051 单片机

—— 1 个时钟 / 机器周期，高速、高可靠，2 路 PWM，8 路高速 8 位 A/D 转换

宏晶科技是新一代增强型8051单片机标准的制定者和领导厂商,致力于提供满足中国市场需求的世界级高性能单片机技术,在业内处于领先地位,销售网络覆盖全国。在高品质的基础上,以极低的价格和完善的服务赢得了客户的长期信赖。现全力推出“1个时钟 / 机器周期”的单片机,全面提升 8051 单片机性能。欢迎海内外厂家前来洽谈合作!新客户请直接联系深圳,以获得更好的技术支持与服务。

强烈推荐选择 SOP-16/20/28/32 贴片封装

每片单片机具有全球唯一身份证号码(ID号)



★如选 32-Pin，推荐选 SOP-32，LQFP32 封装

★如果 I/O 口不够用，可以用 2 到 3 根普通 I/O 口线外接 74HC164/165/595（均可级联）来扩展 I/O 口,还可用 A/D 做按键扫描来节省 I/O 口

STC12C5201AD 系列主要性能：

- ●高速：1个时钟 / 机器周期，增强型 8051 内核，速度比普通 8051 快 8～12 倍
- ●宽电压：5.5～3.3V，2.2～3.6V（STC12LE5201AD 系列）
- ●增加第二复位功能脚（可任意调整复位门槛电压，频率<12MHz 时，无需此功能）
- ●增加掉电检测电路(P1.2),可在掉电时,及时将数据保存进EEPROM(正常工作时无需操作EEPROM)
- ●外部低压掉电检测（P1.2/EX_LVD）
- ●低功耗设计：空闲模式，掉电模式（可由外部中断唤醒）
- ●工作频率：0～35MHz，相当于普通 8051：0～420MHz
- ●时钟：外部晶体或内部 RC 振荡器可选，在 ISP 下载编程用户程序时设置
- ● 1K/2K/4K/5K/6K 字节片内 Flash 程序存储器，擦写次数 10 万次以上
- ● 256 字节片内 RAM 数据存储器
- ●芯片内 EEPROM 功能,擦写次数 10 万次以上
- ● ISP / IAP，在系统可编程 / 在应用可编程,无需编程器 / 仿真器
- ● 8 通道,8 位高速 ADC，速度可达 30 万次 / 秒,2 路 PWM 还可当 2 路 D/A 使用
- ● 2 通道捕获 / 比较单元（PWM/PCA/CCP），
  --- 也可用来再实现 2 个定时器或 2 个外部中断（支持上升沿 / 下降沿中断）
- ● 4 个 16 位定时器，兼容普通 8051 的定时器 T0/T1,2 路 PCA 实现 2 个定时器
- ●可编程时钟输出功能，T0 在 P3.4 输出时钟，T1 在 P3.5 输出时钟
- ●硬件看门狗（WDT）
- ●全双工异步串行口(UART),兼容普通 8051 的串口
- ●先进的指令集结构，兼容普通 8051 指令集
  有硬件乘法 / 除法指令
- ●通用 I/O 口（27/23/15 个），复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）
  可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏
  每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA

选择 STC12C5201AD 系列单片机的理由：

★加密性强,无法解密

★超强抗干扰：
1、高抗静电（ESD 保护），整机轻松过 2 万伏静电测试
2、轻松过 4KV 快速脉冲干扰(EFT 测试)
3、宽电压，不怕电源抖动
4、宽温度范围，-40℃～85℃

★ 1 个时钟 / 机器周期，可用低频晶振，大幅降低 EMI
--- 出口欧美的有力保证

★ 超低功耗：
1、掉电模式： 典型功耗 <0.1 μA
2、空闲模式： 典型功耗 1.8mA
3、正常工作模式： 典型功耗 2.7mA - 7mA
4、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★ 在系统可编程,无需编程器,无需仿真器，可远程升级

★ 可送 STC-ISP 下载编程器,1 万片 / 人 / 天

★ 内部集成 MAX810 专用复位电路，原复位电路可以保留，也可以不用，不用时 RESET 脚接 1K 电阻到地。

STC micro ™
宏 晶 科 技
8051 单片机全球第一品牌
中国本土 MCU 领航者
新客户请直接联系深圳以获得更好的技术支持和服务

网址：www.MCU-Memory.com 技术支持：13922805190

深 圳	Tel：0755-82948411	82948412	Fax：0755-82944243	82905966
广州办	Tel：020-87501705	85518657	Fax：020-85517881	
上海办	Tel：021-53560136	53560138	Fax：021-53080587	
北京办	Tel：010-62538687	62634001	Fax：010-62538683	

超强抗干扰 无法解密 宏晶新一代8051单片机

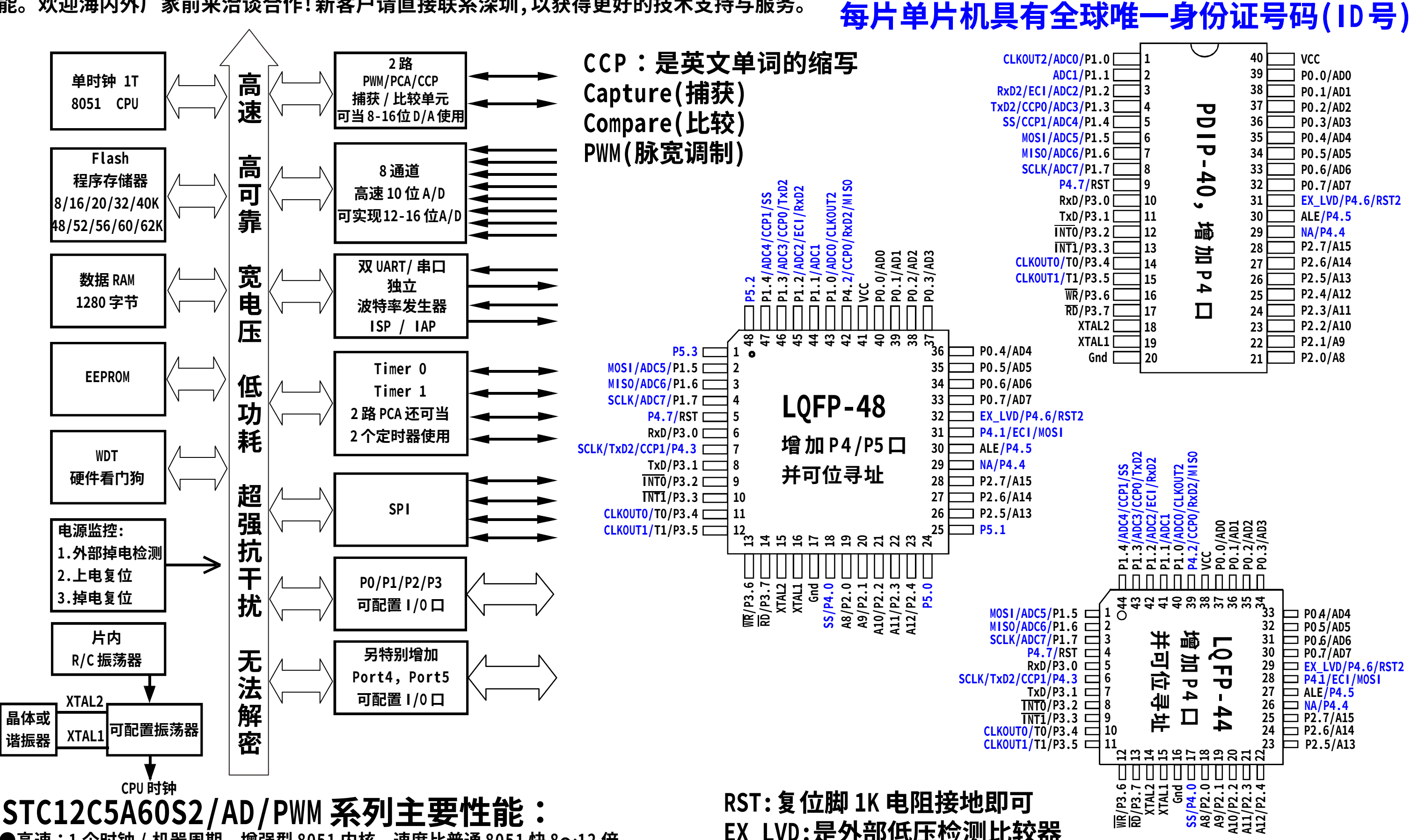
—— 8051 单片机全球第一品牌，中国 MCU 领航者

宏晶 STC12C5A60S2 系列 1T 8051 单片机

—— 1 个时钟 / 机器周期，高速、高可靠，2 路 PWM，8 路高速 A/D 转换

宏晶科技是新一代增强型8051单片机标准的制定者和领导厂商,致力于提供满足中国市场需求的世界级高性能单片机技术,在业内处于领先地位,销售网络覆盖全国。在高品质的基础上,以极低的价格和完善的服务赢得了客户的长期信赖。现全力推出“1 个时钟 / 机器周期”的单片机,全面提升 8051 单片机性能。欢迎海内外厂家前来洽谈合作!新客户请直接联系深圳,以获得更好的技术支持与服务。

每片单片机具有全球唯一身份证号码(ID 号)



STC12C5A60S2/AD/PWM 系列主要性能：

- 高速：1 个时钟 / 机器周期，增强型 8051 内核，速度比普通 8051 快 8～12 倍
- 宽电压：5.5～3.3V，2.2～3.6V（STC12LE5A60S2 系列）
- 增加第二复位功能脚（可任意调整复位门槛电压，频率<12MHz 时，无需此功能）
- 增加掉电检测电路(P4.6),可在掉电时,及时将数据保存进EEPROM,正常工作时无需操作EEPROM
- 低功耗设计：空闲模式，掉电模式（可由外部中断唤醒）
- 工作频率：0～35MHz，相当于普通 8051：0～420MHz
- 时钟：外部晶体或内部 RC 振荡器可选，在 ISP 下载编程用户程序时设置
- 8/16/20/32/40/48/52/56/60/62K 字节片内 Flash 程序存储器，擦写次数 10 万次以上
- 1280 字节片内 RAM 数据存储器
- 芯片内 EEPROM 功能,擦写次数 10 万次以上
- ISP / IAP，在系统可编程 / 在应用可编程,无需编程器 / 仿真器
- 8 通道,10 位高速 ADC，速度可达 25 万次 / 秒,2 路 PWM 还可当 2 路 D/A 使用
- 2 通道捕获 / 比较单元（PWM/PCA/CCP），
  --- 也可用来再实现 2 个定时器或 2 个外部中断（支持上升沿 / 下降沿中断）
- 4 个 16 位定时器，兼容普通 8051 的定时器 T0/T1,2 路 PCA 实现 2 个定时器
- 可编程时钟输出功能，T0 在 P3.4 输出时钟，T1 在 P3.5 输出时钟,BRT 在 P1.0 输出时钟
- 硬件看门狗（WDT）
- 全双工异步串行口(UART),兼容普通 8051 的串口
- 先进的指令集结构，兼容普通 8051 指令集
  有硬件乘法 / 除法指令
- 通用 I/O 口（36/40/44 个），复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）
  可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏
  每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA

RST:复位脚 1K 电阻接地即可
EX_LVD:是外部低压检测比较器
不用的 I/O 口:浮空即可
使用 LQFP48 封装时,最多有 44 个 I/O 口
使用 LQFP44 封装时,最多有 40 个 I/O 口
使用 PDIP40 封装时,最多有 36 个 I/O 口

选择 STC12C5A60S2/AD/PWM 系列单片机的理由：

★ 加密性强,无法解密
★ 超强抗干扰，整机轻松过 2 万伏静电测试
★ 速度快,1 个时钟 / 机器周期，可用低频晶振，大幅降低 EMI
　--- 出口欧美的有力保证
★ 超低功耗:
　掉电模式可由外部中断唤醒，适用于电池
　供电系统，如水表、气表、便携设备等。
★ 输入 / 输出口多,最多有 44 个 I/O 口,A/D 做按键扫描还可以节省很多 I/O
★ 在系统可编程,无需编程器,无需仿真器，可远程升级
★ 可送 STC-ISP 下载编程器,1 万片 / 人 / 天
★ 内部集成 MAX810 专用复位电路，原复位电路可以保留，
　也可以不用，不用时 RESET 脚接 1K 电阻到地。

目录

第二章 STC12 系列单片机总体介绍

2.1.1 STC12C5201AD系列 1T 单片机简介

STC12C5201AD 系列单片机是宏晶科技生产的单时钟 / 机器周期(1T)的单片机，是高速 / 低功耗 / 超强抗干扰的新一代 8051 单片机，指令代码完全兼容传统 8051,但速度快 8-12 倍。内部集成 MAX810 专用复位电路,2 路 PWM,8 路高速 8 位 A/D 转换(300K/S),针对电机控制，强干扰场合。

1. 增强型 8051 CPU，1T，单时钟 / 机器周期，指令代码完全兼容传统 8051
2. 工作电压：
   STC12C5201AD 系列工作电压： 5.5V - 3.3V（5V 单片机）
   STC12LE5201AD 系列工作电压：3.6V - 2.2V（3V 单片机）
3. 工作频率范围：0 - 35MHz，相当于普通 8051 的 0～420MHz
4. 用户应用程序空间 1K / 2K / 4K / 5K / 6K 字节......
5. 片上集成 256 字节 RAM
6. 通用 I/O 口（27/23/15/13/11 个），复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）
   可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏
   每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不要超过 55mA
7. ISP（在系统可编程）/ IAP（在应用可编程），无需专用编程器，无需专用仿真器
   可通过串口（P3.0/P3.1）直接下载用户程序，数秒即可完成一片
8. 有 EEPROM 功能
9. 看门狗
10. 内部集成 MAX810 专用复位电路（外部晶体 20M 以下时，复位脚可直接 1K 电阻到地）
11. 内置一个掉电检测电路，在 P1.2 口有一个低压门槛比较器
    5V 单片机为 1.32V，误差为 +/-5%,3.3V 单片机为 1.30V，误差为 +/-3%
12. 时钟源：外部高精度晶体 / 时钟，内部 R/C 振荡器(温漂为 +/-5% 到 +/-10% 以内)
    用户在下载用户程序时，可选择是使用内部 R/C 振荡器还是外部晶体 / 时钟
    常温下内部 R/C 振荡器频率为：5.0V 单片机为： 11MHz ～ 15.5MHz
    3.3V 单片机为： 8MHz ～ 12MHz
    精度要求不高时，可选择使用内部时钟，但因为有制造误差和温漂，以实际测试为准
13. 共 4 个 16 位定时器
    两个与传统 8051 兼容的定时器 / 计数器,16 位定时器 T0 和 T1
    再加上 2 路 PCA 模块可再实现 2 个 16 位定时器
14. 2 个时钟输出口，可由 T0 的溢出在 P3.4/T0 输出时钟，可由 T1 的溢出在 P3.5/T1 输出时钟
15. 外部中断 I/O 口 6 路,传统的下降沿中断或低电平触发中断,并新增支持上升沿中断的 PCA 模块，
    Power Down 模式可由外部中断唤醒，
    INT0/P3.2，INT1/P3.3，T0/P3.4，T1/P3.5，RxD/P3.0，PCA0/P3.7，PCA1/P3.5
16. PWM(2 路）/PCA（可编程计数器阵列,2 路）
    --- 也可用来当 2 路 D/A 使用
    --- 也可用来再实现 2 个定时器
    --- 也可用来再实现 2 个外部中断(上升沿中断 / 下降沿中断均可分别或同时支持)
17. A/D 转换，8 位精度 ADC，共 8 路，转换速度可达 300K/S(每秒钟 30 万次)
18. 通用全双工异步串行口(UART)，由于 STC12 系列是高速的 8051，可再用定时器或 PCA 软件实现多串口
19. 工作温度范围： -40 - +85℃(工业级) / 0 - 75℃(商业级)
20. 封装：LQFP-32,SOP-32/28/20/16，SKDIP-28，PDIP-20/18/16，LSSOP-20(超小封状 6.4mm × 6.4mm)
    LQFP/SOP32 有 27 个 I/O 口，SOP28/SKDIP28 有 23 个 I/O 口，SOP20/LSSOP20/PDIP20 有 15 个 I/O 口，DIP18 有 13 个 I/O 口，SOP16/DIP16 有 11 个 I/O 口。I/O 口不够时，可用 2 到 3 根普通 I/O 口线外接 74HC164/165/595（均可级联）来扩展 I/O 口，
    还可用 A/D 做按键扫描来节省 I/O 口，或用双 CPU,三线通信，还多了串口。

2.1.2 STC12C5A60S2系列 1T 单片机简介

STC12C5A60S2/AD/PWM系列单片机是宏晶科技生产的单时钟/机器周期(1T)的单片机，是高速/低功耗/超强抗干扰的新一代8051单片机，指令代码完全兼容传统8051,但速度快8-12倍。内部集成MAX810专用复位电路,2路PWM,8路高速10位A/D转换(250K/S),针对电机控制，强干扰场合。

1. 增强型 8051 CPU，1T，单时钟/机器周期，指令代码完全兼容传统8051
2. 工作电压：
   STC12C5A60S2系列工作电压： 5.5V - 3.3V（5V单片机）
   STC12LE5A60S2系列工作电压：3.6V - 2.2V（3V单片机）
3. 工作频率范围：0 - 35MHz，相当于普通8051的 0～420MHz
4. 用户应用程序空间 8K /16K / 20K / 32K / 40K / 48K / 52K / 60K / 62K字节......
5. 片上集成 1280 字节 RAM
6. 通用I/O口（36/40/44个），复位后为： 准双向口/弱上拉（普通8051传统I/O口）
   可设置成四种模式：准双向口/弱上拉，推挽/强上拉，仅为输入/高阻，开漏
   每个I/O口驱动能力均可达到20mA，但整个芯片最大不要超过55mA
7. ISP（在系统可编程）/IAP（在应用可编程），无需专用编程器，无需专用仿真器
   可通过串口（P3.0/P3.1）直接下载用户程序，数秒即可完成一片
8. 有EEPROM 功能(STC12C5A62S2/AD/PWM无内部EEPROM)
9. 看门狗
10. 内部集成MAX810专用复位电路（外部晶体12M以下时，复位脚可直接1K电阻到地）
11. 外部掉电检测电路：在P4.6口有一个低压门槛比较器
    5V单片机为1.32V，误差为+/-5%,3.3V单片机为1.30V，误差为+/-3%
12. 时钟源：外部高精度晶体/时钟，内部R/C 振荡器(温漂为+/-5%到+/-10%以内)
    用户在下载用户程序时，可选择是使用内部R/C振荡器还是外部晶体/时钟
    常温下内部R/C 振荡器频率为：5.0V单片机为： 11MHz ～ 15.5MHz
    3.3V单片机为： 8MHz ～ 12MHz
    精度要求不高时，可选择使用内部时钟，但因为有制造误差和温漂，以实际测试为准
13. 共4个16位定时器
    两个与传统8051兼容的定时器/计数器,16位定时器T0和T1，没有定时器2，但有独立波特率发生器做串行通讯的波特率发生器
    再加上2路PCA模块可再实现2个16位定时器
14. 2个时钟输出口，可由T0的溢出在P3.4/T0输出时钟，可由T1的溢出在P3.5/T1输出时钟
15. 外部中断I/O口7路,传统的下降沿中断或低电平触发中断,并新增支持上升沿中断的PCA模块,
    Power Down模式可由外部中断唤醒，
    INT0/P3.2，INT1/P3.3，T0/P3.4，T1/P3.5，RxD/P3.0，
    CCP0/P1.3(也可通过寄存器设置到P4.2)，CCP1/P1.4 (也可通过寄存器设置到P4.3)
16. PWM(2路）/PCA（可编程计数器阵列,2路）
    --- 也可用来当2路D/A使用
    --- 也可用来再实现2个定时器
    --- 也可用来再实现2个外部中断(上升沿中断/下降沿中断均可分别或同时支持)
17. A/D转换，10位精度ADC，共8路，转换速度可达250K/S(每秒钟25万次)
18. 通用全双工异步串行口(UART)，由于STC12系列是高速的8051，可再用定时器或PCA软件实现多串口
19. STC12C5A60S2系列有双串口，后缀有S2标志的才有双串口，RxD2/P1.2(可通过寄存器设置到P4.2)，TxD2/P1.3(可通过寄存器设置到P4.3)
20. 工作温度范围： -40 - +85℃(工业级) / 0 - 75℃(商业级)
21. 封装：PDIP-40,LQFP-44,LQFP-48
    I/O口不够时，可用2到3根普通I/O口线外接 74HC164/165/595（均可级联）来扩展I/O口，还可用A/D 做按键扫描来节省I/O口，或用双CPU,三线通信，还多了串口。

2.2 STC12系列单片机选型一览表

2.2.1 STC12C5201AD系列单片机选型一览表

型号	工作电压(V)	Flash程序存储器字节	SRAM字节	定时器T0 T1	PCA定时器	UART串口	DPTR	EEPROM	PCA 16位PWM 8位	A/D 8路	I/O	看门狗	内置复位	内部低压检测	封装16-Pin	封装18-Pin	封装20-Pin	封装28-Pin	封装32-Pin
STC12C5201AD/PWM系列单片机选型一览																			
STC12C5201	5.5 - 3.3	1K	256	有		有	1				11/13/15	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP		
STC12C5201PWM	5.5 - 3.3	1K	256	有	2	有	1	有	2路		11/13/15	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP		
STC12C5201AD	5.5 - 3.3	1K	256	有	2	有	1	有	2路	8位	11/13/15	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP		
STC12C5202	5.5 - 3.3	2K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5202PWM	5.5 - 3.3	2K	256	有	2	有	1	有	2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5202AD	5.5 - 3.3	2K	256	有	2	有	1	有	2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5204	5.5 - 3.3	4K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5204PWM	5.5 - 3.3	4K	256	有	2	有	1	有	2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5204AD	5.5 - 3.3	4K	256	有	2	有	1	有	2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5205	5.5 - 3.3	5K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5205PWM	5.5 - 3.3	5K	256	有	2	有	1	有	2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5205AD	5.5 - 3.3	5K	256	有	2	有	1	有	2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5206	5.5 - 3.3	6K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	可在应用程序区修改应用程序		
STC12C5206PWM	5.5 - 3.3	6K	256	有	2	有	1		2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	可在应用程序区修改应用程序		
STC12C5206AD	5.5 - 3.3	6K	256	有	2	有	1		2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	可在应用程序区修改应用程序		
STC12LE5201AD/PWM系列单片机选型一览																			
STC12LE5201	3.6 - 2.2	1K	256	有		有	1				11/13/15	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP		
STC12LE5201PWM	3.6 - 2.2	1K	256	有	2	有	1	有	2路		11/13/15	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP		
STC12LE5201AD	3.6 - 2.2	1K	256	有	2	有	1	有	2路	8位	11/13/15	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP		
STC12LE5202	3.6 - 2.2	2K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5202PWM	3.6 - 2.2	2K	256	有	2	有	1	有	2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5202AD	3.6 - 2.2	2K	256	有	2	有	1	有	2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5204	3.6 - 2.2	4K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5204PWM	3.6 - 2.2	4K	256	有	2	有	1	有	2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5204AD	3.6 - 2.2	4K	256	有	2	有	1	有	2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5205	3.6 - 2.2	5K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5205PWM	3.6 - 2.2	5K	256	有	2	有	1	有	2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5205AD	3.6 - 2.2	5K	256	有	2	有	1	有	2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	SOP/LSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5206	3.6 - 2.2	6K	256	有		有	1				11/13/15/23/27	有	有	有	SOP/DIP	DIP	可在应用程序区修改应用程序		
STC12LE5206PWM	3.6 - 2.2	6K	256	有	2	有	1		2路		11/13/15/23/27	有	有	有	SOP/DIP	DIP	可在应用程序区修改应用程序		
STC12LE5206AD	3.6 - 2.2	6K	256	有	2	有	1		2路	8位	11/13/15/23/27	有	有	有	SOP/DIP	DIP	可在应用程序区修改应用程序		

2.2.2 STC12C5A60AD系列单片机选型一览表

型号	工作电压(V)	Flash程序存储器字节	SRAM字节	定时器T0 T1	PCA定时器	UART	独立波特率发生器	DPTR	EEPROM	PCA 16位PWM 8位	A/D 8路	I/O	看门狗	内置复位	外部低压检测	封装40-Pin	封装44-Pin	封装48-Pin
STC12C5A60AD系列单片机选型一览																		
STC12C5A08PWM	5.5 - 3.3	8K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A08AD	5.5 - 3.3	8K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A08S2	5.5 - 3.3	8K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A16PWM	5.5 - 3.3	16K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A16AD	5.5 - 3.3	16K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A16S2	5.5 - 3.3	16K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A20PWM	5.5 - 3.3	20K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A20AD	5.5 - 3.3	20K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A20S2	5.5 - 3.3	20K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A32PWM	5.5 - 3.3	32K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A32AD	5.5 - 3.3	32K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A32S2	5.5 - 3.3	32K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A40PWM	5.5 - 3.3	40K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A40AD	5.5 - 3.3	40K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A40S2	5.5 - 3.3	40K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A48PWM	5.5 - 3.3	48K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A48AD	5.5 - 3.3	48K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A48S2	5.5 - 3.3	48K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A52PWM	5.5 - 3.3	52K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A52AD	5.5 - 3.3	52K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A52S2	5.5 - 3.3	52K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A56PWM	5.5 - 3.3	56K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A56AD	5.5 - 3.3	56K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A56S2	5.5 - 3.3	56K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A60PWM	5.5 - 3.3	60K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A60AD	5.5 - 3.3	60K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A60S2	5.5 - 3.3	60K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A62PWM	5.5 - 3.3	62K	1280	有	2	1	有	2		2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A62AD	5.5 - 3.3	62K	1280	有	2	1	有	2		2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12C5A62S2	5.5 - 3.3	62K	1280	有	2	2	有	2		2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48

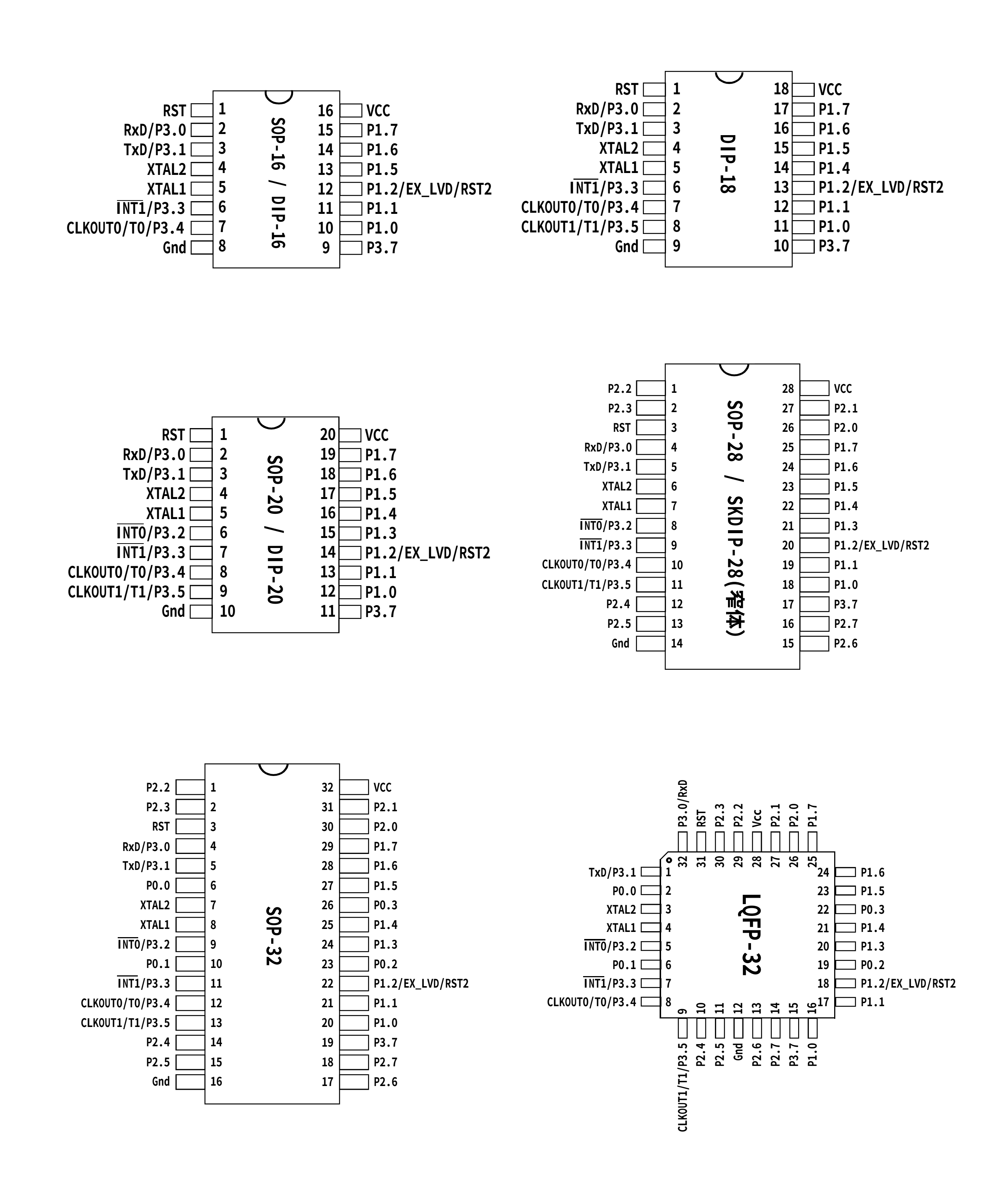
型 号	工作电压(V)	Flash程序存储器字节	SRAM字节	定时器T0 T1	PCA定时器	UART	独立波特率发生器	DPTR	EEPROM	PCA 16位PWM 8位	A/D 8路	I/O	看门狗	内置复位	外部低压检测	封装40-Pin	封装44-Pin	封装48-Pin
STC12C5A60AD系列单片机选型一览																		
STC12LE5A08PWM	3.6 - 2.2	8K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A08AD	3.6 - 2.2	8K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A08S2	3.6 - 2.2	8K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A16PWM	3.6 - 2.2	16K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A16AD	3.6 - 2.2	16K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A16S2	3.6 - 2.2	16K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A20PWM	3.6 - 2.2	20K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A20AD	3.6 - 2.2	20K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A20S2	3.6 - 2.2	20K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A32PWM	3.6 - 2.2	32K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A32AD	3.6 - 2.2	32K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A32S2	3.6 - 2.2	32K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A40PWM	3.6 - 2.2	40K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A40AD	3.6 - 2.2	40K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A40S2	3.6 - 2.2	40K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A48PWM	3.6 - 2.2	48K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A48AD	3.6 - 2.2	48K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A48S2	3.6 - 2.2	48K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A52PWM	3.6 - 2.2	52K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A52AD	3.6 - 2.2	52K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A52S2	3.6 - 2.2	52K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A56PWM	3.6 - 2.2	56K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A56AD	3.6 - 2.2	56K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A56S2	3.6 - 2.2	56K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A60PWM	3.6 - 2.2	60K	1280	有	2	1	有	2	有	2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A60AD	3.6 - 2.2	60K	1280	有	2	1	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A60S2	3.6 - 2.2	60K	1280	有	2	2	有	2	有	2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A62PWM	3.6 - 2.2	62K	1280	有	2	1	有	2		2路		36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A62AD	3.6 - 2.2	62K	1280	有	2	1	有	2		2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48
STC12LE5A62S2	3.6 - 2.2	62K	1280	有	2	2	有	2		2路	10位	36/40/44	有	有	有	PDIP40	LQFP44	LQFP48

2.3 STC12C5201AD系列单片机管脚图及封装尺寸图

2.3.1 管脚图(所有封装形式均满足欧盟 RoHS 要求，LQFP-32 采用 Green 标准生产)

强烈推荐选择 SOP-16/20/28/32 贴片封装,传统的插件 DIP 封装稳定供货

STC12C5201 系列管脚图

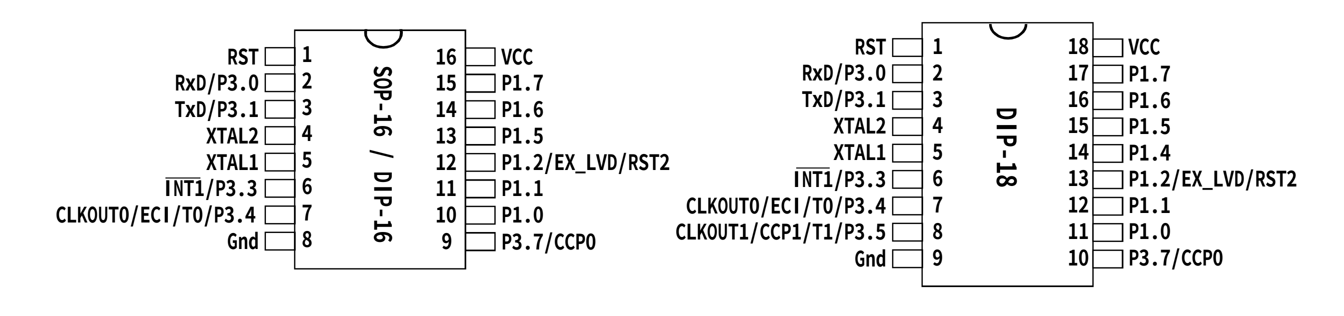


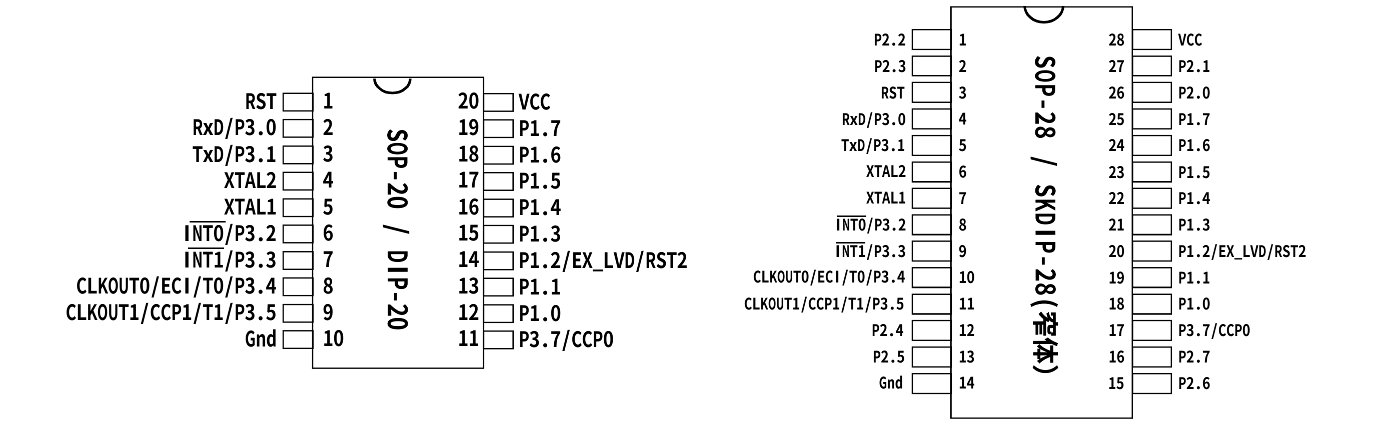
STC12C5202 系列(无 A/D 转换,无 PWM 功能，无内部 EEPROM)

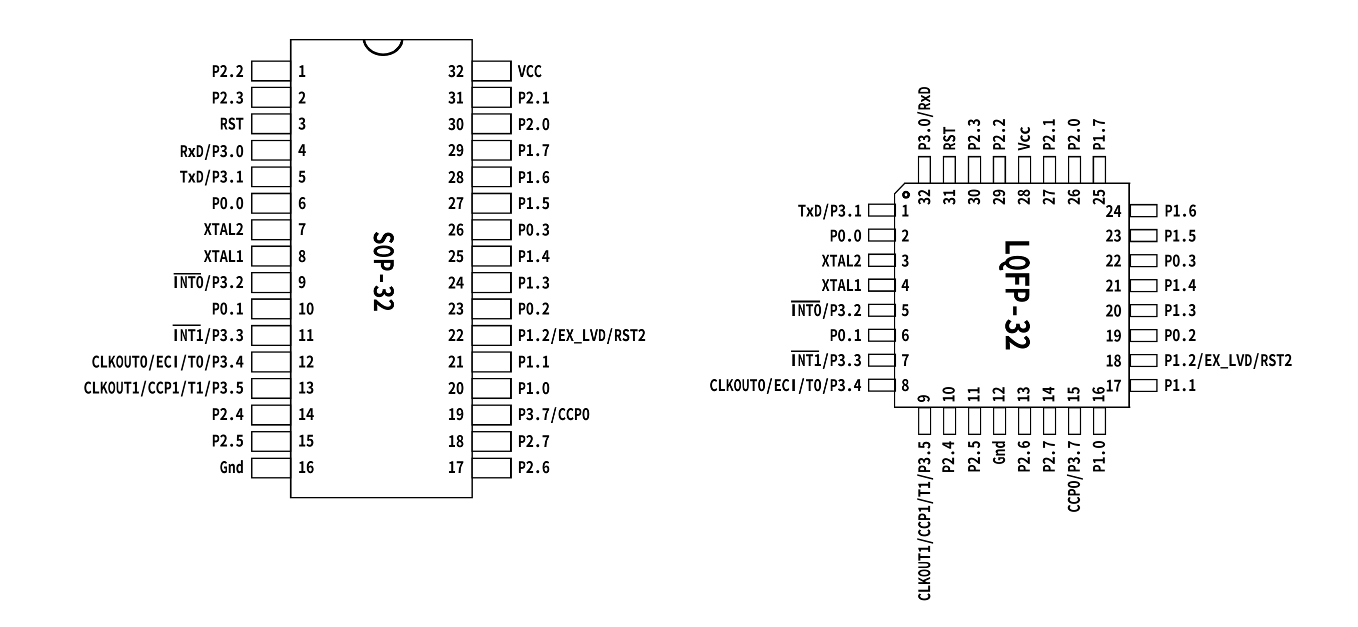
STC12LE5202 系列(无 A/D 转换,无 PWM 功能，无内部 EEPROM)

STC12C5201PWM 系列管脚图

CCP：是英文单词的缩写
Capture(捕获),Compare(比较),PWM(脉宽调制)



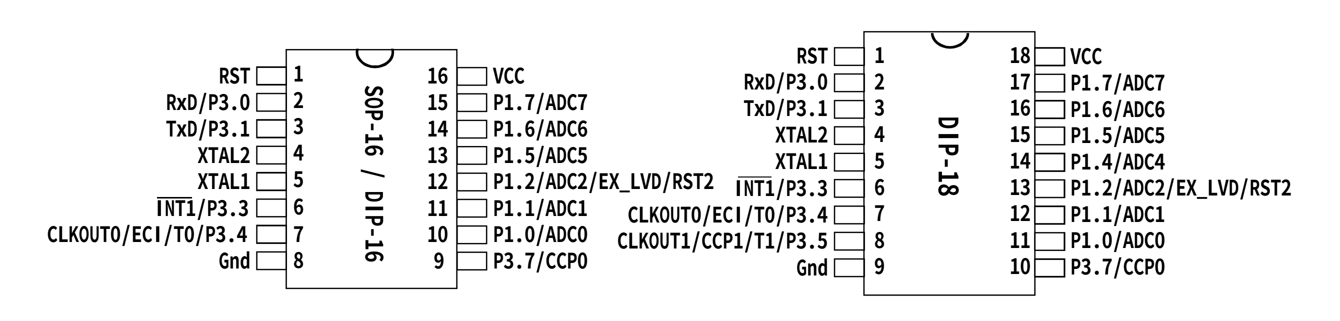


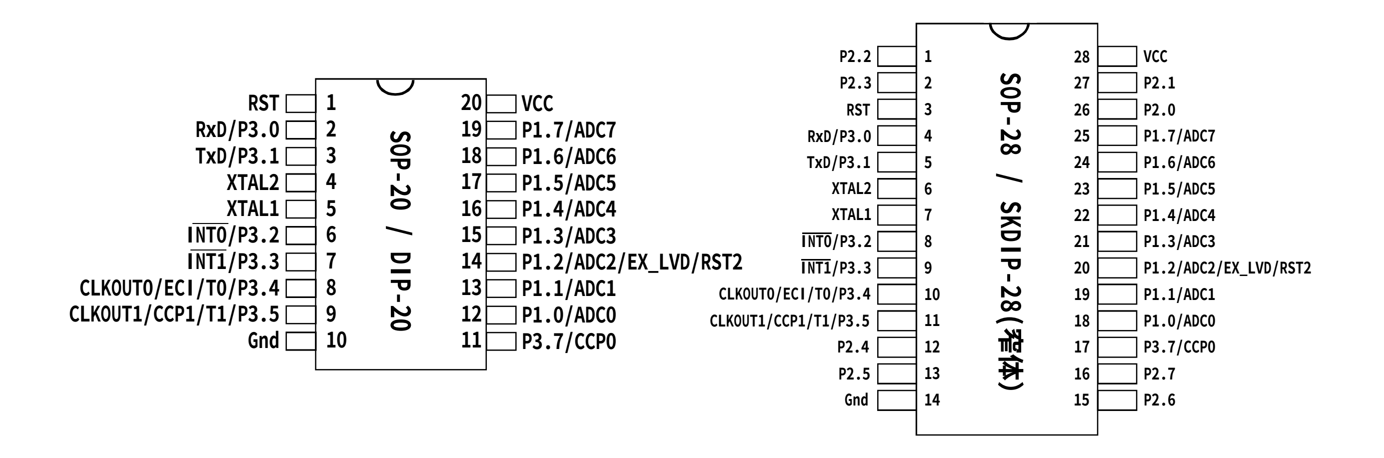


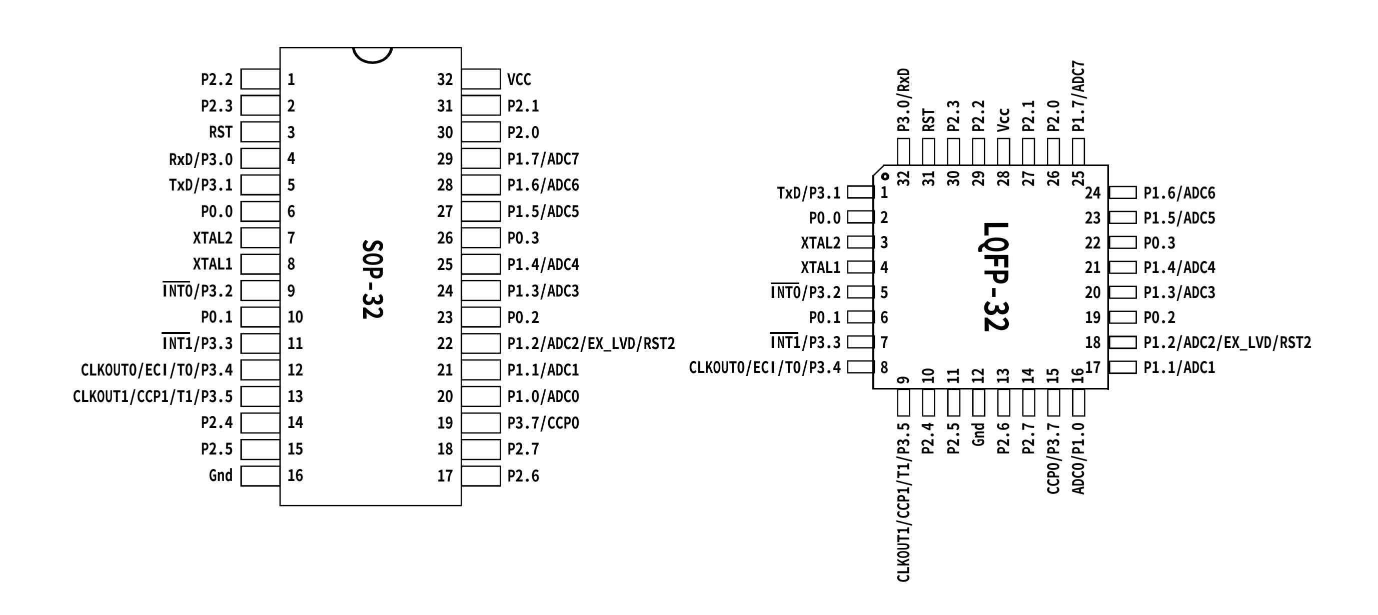
STC12C5202PWM 系列(无 A/D 转换,有 PWM 功能，有内部 EEPROM)
STC12LE5202PWM 系列(无 A/D 转换,有 PWM 功能，有内部 EEPROM)

STC12C5201AD 系列管脚图

CCP：是英文单词的缩写
Capture(捕获),Compare(比较),PWM(脉宽调制)





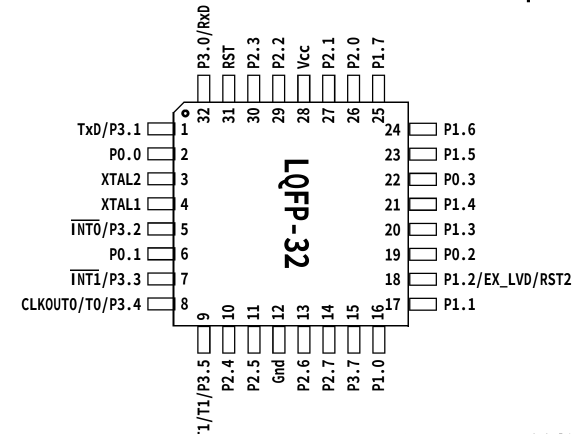


STC12C5202AD 系列(有 A/D 转换,有 PWM 功能，有内部 EEPROM)
STC12LE5202AD 系列(有 A/D 转换,有 PWM 功能，有内部 EEPROM)

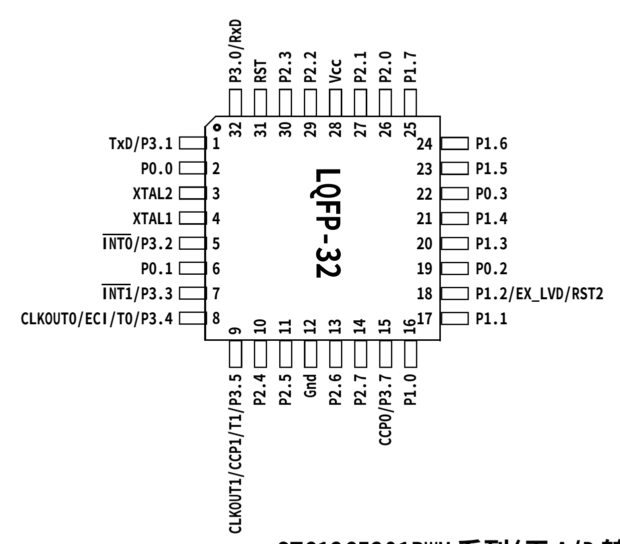
LQFP-32 管脚图

长 x 宽 = 9mm x 9mm, 高 < 1.6mm

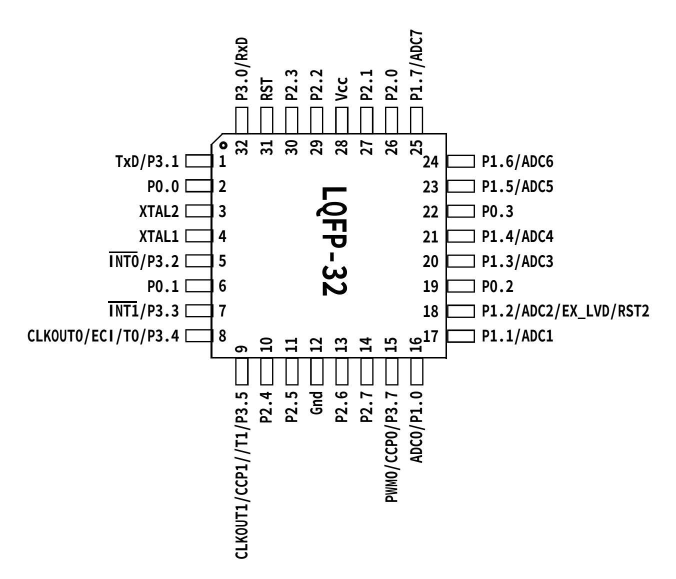
CCP：是英文单词的缩写
Capture(捕获),Compare(比较),PWM(脉宽调制)



STC12C5201 系列(无 A/D 转换,无 PWM 功能,无内部 EEPROM),32-Pin
STC12LE5201 系列(无 A/D 转换,无 PWM 功能,无内部 EEPROM),32-Pin



STC12C5201PWM 系列(无 A/D 转换,有 PWM 功能,有内部 EEPROM),32-Pin
STC12LE5201PWM 系列(无 A/D 转换,有 PWM 功能,有内部 EEPROM),32-Pin



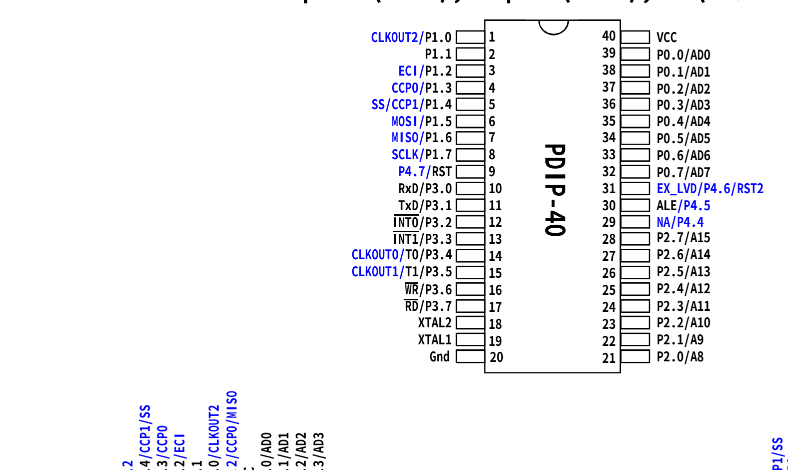
STC12C5201AD 系列(有 A/D 转换,PWM 功能,有内部 EEPROM),32-Pin
STC12LE5201AD 系列(有 A/D 转换,PWM 功能,有内部 EEPROM),32-Pin

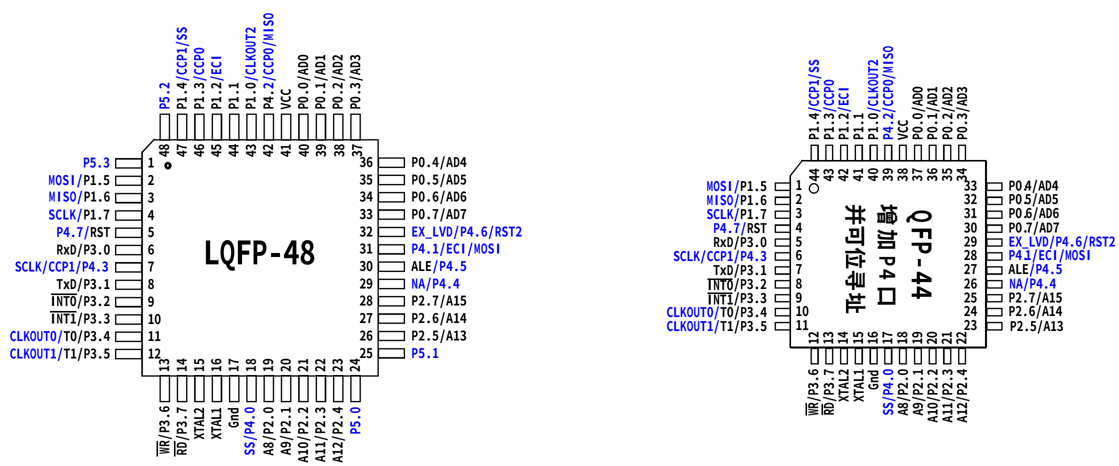
2.3.2 STC12C5A60S2系列单片机管脚图

STC12C5A60系列单片机管脚图

CCP：是英文单词的缩写

Capture(捕获),Compare(比较),PWM(脉宽调制)





STC12C5A60PWM 系列(无第二串口,无 A/D 转换,有 PWM/PCA 功能，有内部 EEPROM)

STC12LE5A60PWM 系列(无第二串口,无 A/D 转换,有 PWM/PCA 功能，有内部 EEPROM)

由 P4SW 寄存器设置(NA/P4.4，ALE/P4.5，EX_LVD/P4.6)三个端口的第二功能

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P4SW	BBh	Port - 4 switch		LVD_P4.6	ALE_P4.5	NA_P4.4					x000,xxxx

NA/P4.4: 0,复位后 P4SW.4 = 0,NA/P4.4 脚是弱上拉，无任何功能

1,通过设置 P4SW.4 = 1,将NA/P4.4脚设置成 I/O口(P4.4)

ALE/P4.5: 0,复位后P4SW.5 = 0,ALE/P4.5脚是ALE信号,只有在用MOVX指令访问片外扩展器件时才有信号输出

1,通过设置 P4SW.5 = 1,将 ALE/P4.5 脚设置成 I/O 口(P4.5)

EX_LVD/P4.6: 0,复位后 P4SW.6 = 0,EX_LVD/P4.6 是外部低压检测脚，可使用查询方式或设置成中断来检测

1,通过设置P4SW.6 = 1将EX_LVD/P4.6脚设置成I/O口(P4.6)

在 ISP 烧录程序时设置 RST/P4.7 的第二功能

RST/P4.7 在 ISP 烧录程序时选择是复位脚还是 P4.7 口，如设置成 P4.7 口，必须使用外部时钟。

由 AUXR1 寄存器设置(PCA/PWM/SPI/UART2)是在 P1 口还是在 P4 口

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1	A2h	Auxiliary register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0

PCA_P4: 0,复位后AUXR1.6 = 0,PCA/PWM在P1口

1,通过设置 AUXR1.6 = 1,将 PCA/PWM 从 P1 口切换到 P4 口

SPI_P4: 0,复位后AUXR1.5 = 0,SPI在P1口

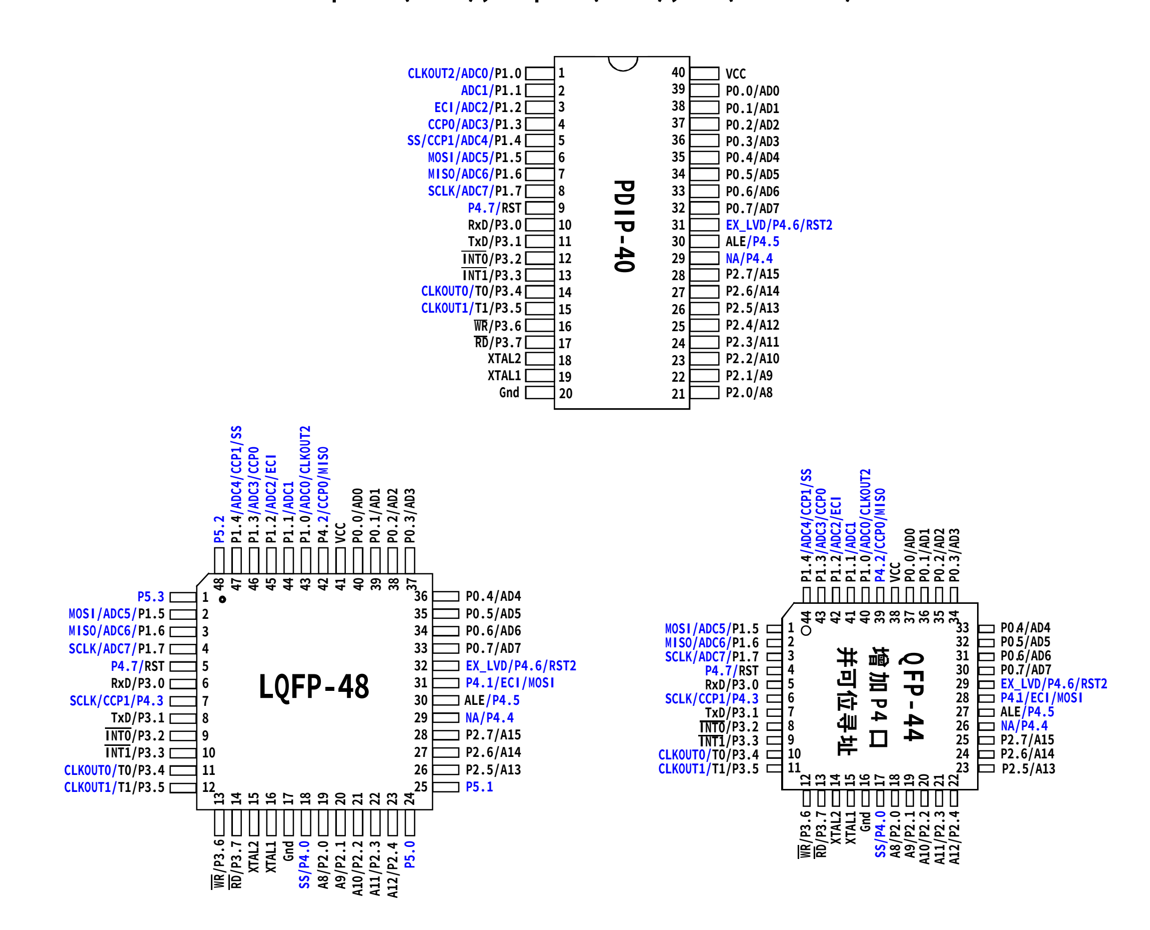
1,通过设置 AUXR1.5 = 1, 将 SPI 从 P1 口切换到 P4 口

S2_P4: 0,复位后 AUXR1.4 = 0,UART2/ 串口2在P1口(仅针对双串口单片机有效)

1,通过设置 AUXR1.4 = 1, 将 UART2/ 串口 2 从 P1 口切换到 P4 口(仅针对双串口单片机有效)

STC12C5A60AD 系列单片机管脚图

CCP：是英文单词的缩写
Capture(捕获),Compare(比较),PWM(脉宽调制)



STC12C5A60AD 系列(无第二串口,有 A/D 转换,有 PWM/PCA 功能，有内部 EEPROM)
STC12LE5A60AD 系列(无第二串口,有 A/D 转换,有 PWM/PCA 功能，有内部 EEPROM)

由 P4SW 寄存器设置(NA/P4.4，ALE/P4.5，EX_LVD/P4.6)三个端口的第二功能

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P4SW	BBh	Port - 4 switch		LVD_P4.6	ALE_P4.5	NA_P4.4					x000,xxxx

NA/P4.4: 0,复位后 P4SW.4 = 0,NA/P4.4 脚是弱上拉，无任何功能
1,通过设置 P4SW.4 = 1,将 NA/P4.4 脚设置成 I/O 口(P4.4)

ALE/P4.5: 0,复位后 P4SW.5 = 0,ALE/P4.5 脚是 ALE 信号,只有在用 MOVX 指令访问片外扩展器件时才有信号输出
1，通过设置 P4SW.5 = 1,将 ALE/P4.5 脚设置成 I/O 口(P4.5)

EX_LVD/P4.6: 0，复位后 P4SW.6 = 0,EX_LVD/P4.6 是外部低压检测脚，可使用查询方式或设置成中断来检测
1,通过设置 P4SW.6 = 1 将 EX_LVD/P4.6 脚设置成 I/O 口(P4.6)

在 ISP 烧录程序时设置 RST/P4.7 的第二功能

RST/P4.7 在 ISP 烧录程序时选择是复位脚还是 P4.7 口，如设置成 P4.7 口，必须使用外部时钟。

由 AUXR1 寄存器设置(PCA/PWM/SPI/UART2)是在 P1 口还是在 P4 口

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1	A2h	Auxiliary register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0

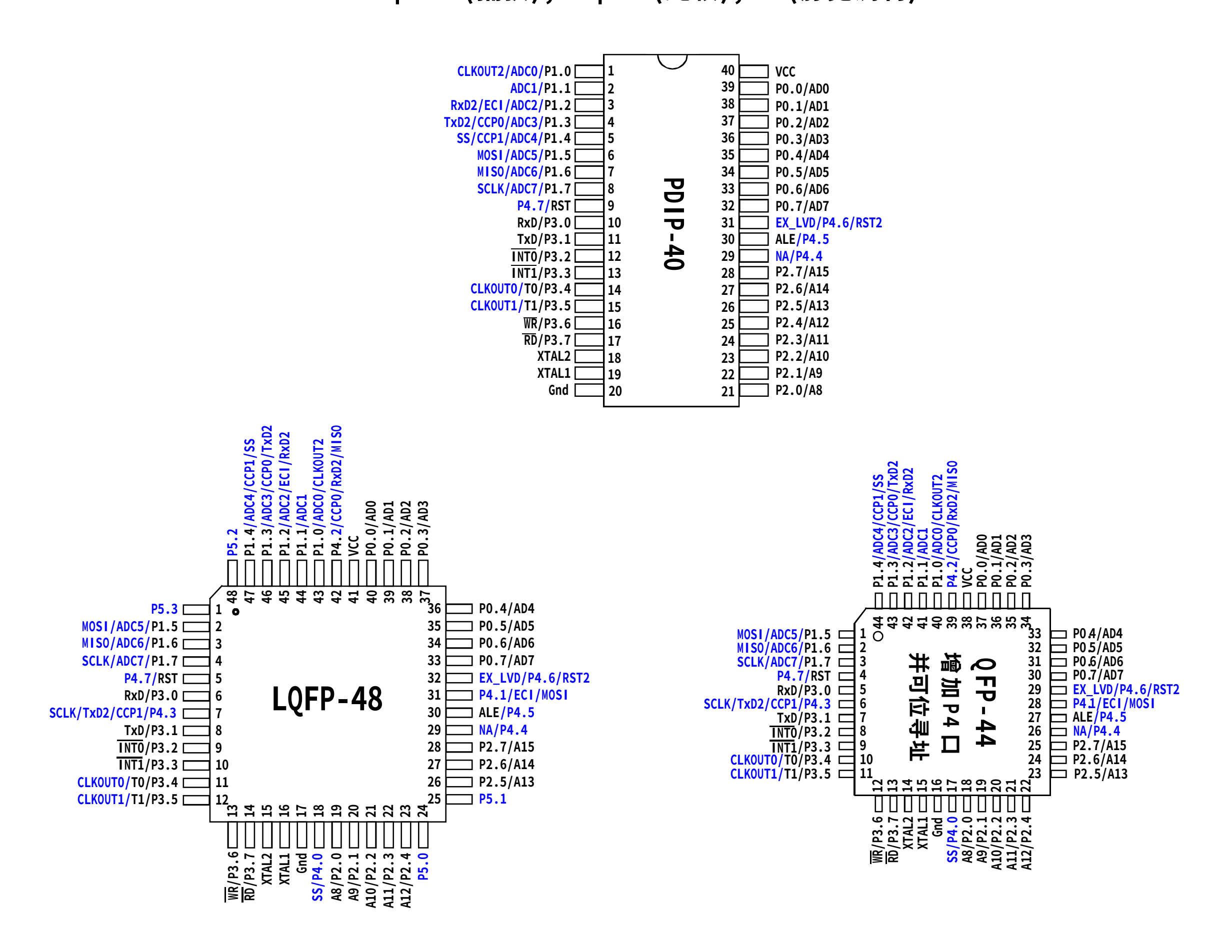
PCA_P4: 0,复位后 AUXR1.6 = 0,PCA/PWM 在 P1 口
1,通过设置 AUXR1.6 = 1,将 PCA/PWM 从 P1 口切换到 P4 口

SPI_P4: 0,复位后 AUXR1.5 = 0,SPI 在 P1 口
1,通过设置 AUXR1.5 = 1，将 SPI 从 P1 口切换到 P4 口

S2_P4: 0,复位后 AUXR1.4 = 0,UART2/串口 2 在 P1 口(仅针对双串口单片机有效)
1,通过设置 AUXR1.4 = 1，将 UART2/串口 2 从 P1 口切换到 P4 口(仅针对双串口单片机有效)

STC12C5A60S2 系列单片机管脚图

CCP：是英文单词的缩写
Capture(捕获),Compare(比较),PWM(脉宽调制)



STC12C5A60S2 系列(有第二串口,有 A/D 转换,有 PWM/PCA 功能，有内部 EEPROM)
STC12LE5A60S2 系列(有第二串口,有 A/D 转换,有 PWM/PCA 功能，有内部 EEPROM)

由 P4SW 寄存器设置(NA/P4.4，ALE/P4.5，EX_LVD/P4.6)三个端口的第二功能

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P4SW	BBh	Port - 4 switch		LVD_P4.6	ALE_P4.5	NA_P4.4					x000,xxxx

NA/P4.4: 0,复位后 P4SW.4 = 0,NA/P4.4 脚是弱上拉，无任何功能
1,通过设置 P4SW.4 = 1,将 NA/P4.4 脚设置成 I/O 口(P4.4)

ALE/P4.5: 0,复位后 P4SW.5 = 0,ALE/P4.5 脚是 ALE 信号,只有在用 MOVX 指令访问片外扩展器件时才有信号输出
1，通过设置 P4SW.5 = 1,将 ALE/P4.5 脚设置成 I/O 口(P4.5)

EX_LVD/P4.6: 0， 复位后 P4SW.6 = 0,EX_LVD/P4.6 是外部低压检测脚，可使用查询方式或设置成中断来检测
1,通过设置 P4SW.6 = 1 将 EX_LVD/P4.6 脚设置成 I/O 口(P4.6)

在 ISP 烧录程序时设置 RST/P4.7 的第二功能

RST/P4.7 在 ISP 烧录程序时选择是复位脚还是 P4.7 口，如设置成 P4.7 口，必须使用外部时钟。

由 AUXR1 寄存器设置(PCA/PWM/SPI/UART2)是在 P1 口还是在 P4 口

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1	A2h	Auxiliary register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0

PCA_P4: 0,复位后 AUXR1.6 = 0,PCA/PWM 在 P1 口
1,通过设置 AUXR1.6 = 1,将 PCA/PWM 从 P1 口切换到 P4 口

SPI_P4: 0,复位后 AUXR1.5 = 0,SPI 在 P1 口
1,通过设置 AUXR1.5 = 1, 将 SPI 从 P1 口切换到 P4 口

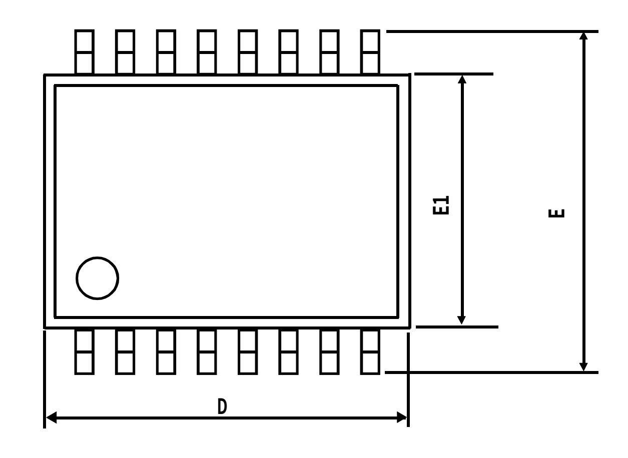
S2_P4: 0,复位后 AUXR1.4 = 0,UART2/ 串口 2 在 P1 口(仅针对双串口单片机有效)
1,通过设置 AUXR1.4 = 1, 将 UART2/ 串口 2 从 P1 口切换到 P4 口(仅针对双串口单片机有效)

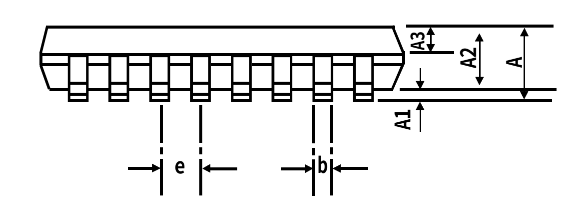
2.4 STC12系列单片机封装尺寸图

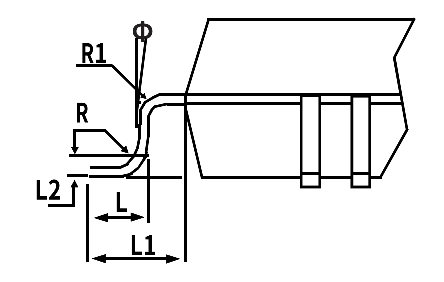
2.4.1 STC12C5202AD系列单片机封装尺寸图

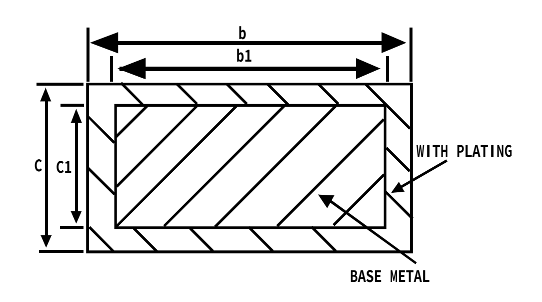
SOP-16 封装尺寸图

16-PIN SMALL OUTLINE PACKAGE (SOP-16)





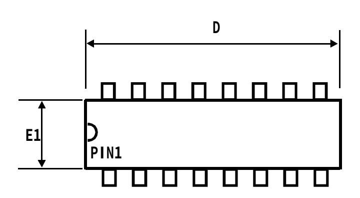


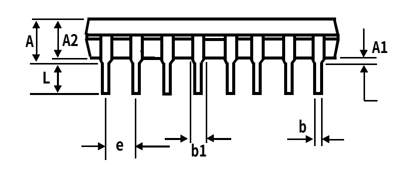


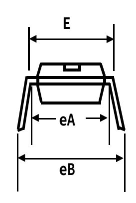
COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	1.35	1.60	1.75
A1	0.10	0.15	0.25
A2	1.25	1.45	1.65
A3	0.55	0.65	0.75
b	0.36	-	0.49
b1	0.35	0.40	0.45
c	0.16	-	0.25
c1	0.15	0.20	0.25
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
L	0.45	0.60	0.80
L1	1.04REF		
L2	0.25BSC		
R	0.07	-	-
R1	0.07	-	-
Φ	6°	8°	10°

PDIP-16 封装尺寸图

Plastic Dual Inline Package (PDIP-16)
Dimensions in Inches and (Millimeters)



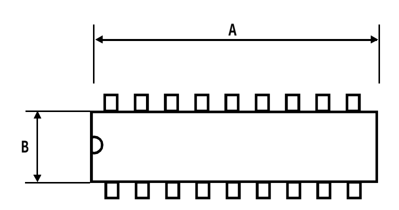


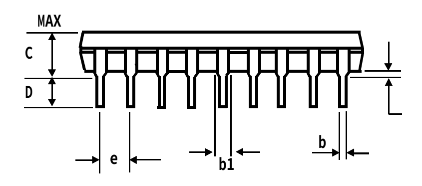


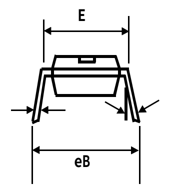
COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	-	-	4.80
A1	0.50	-	-
A2	3.10	3.30	3.50
b	0.38	-	0.55
b1	0.38	0.46	0.51
D	18.95	19.05	19.15
E	7.62	7.87	8.25
E1	6.25	6.35	6.45
e	2.54BSC		
eA	7.62BSC		
eB	7.62	8.80	10.90
L	2.92	3.30	3.81

PDIP-18 封装尺寸图

Plastic Dual Inline Package (PDIP-18)
Dimensions in Inches and (Millimeters)



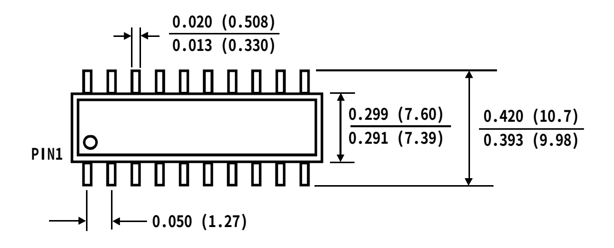


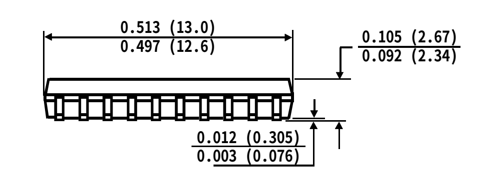


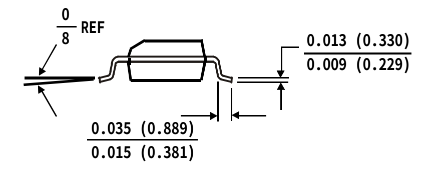
COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	22.72	-	23.23
B	6.10	-	6.60
C	3.18	-	3.43
D	3.18	-	3.69
e	-	2.54	-
b	0.41	-	0.51
b1	1.27	-	1.78
E	7.49	-	8.00
eB	8.51	-	9.52

SOP-20 封装尺寸图

Plastic Gull WIng Small Outline (SOIC-20 / SOP-20)
Dimensions in Inches and (Millimeters)

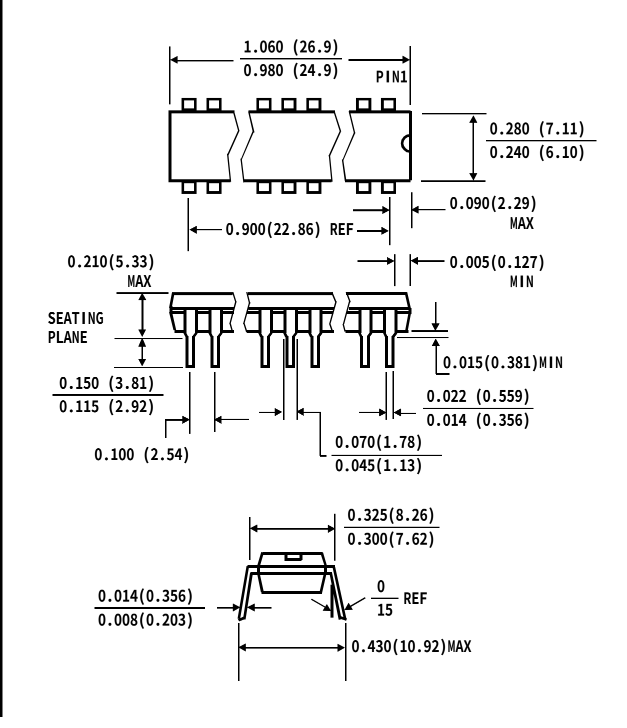






PDIP-20 封装尺寸图

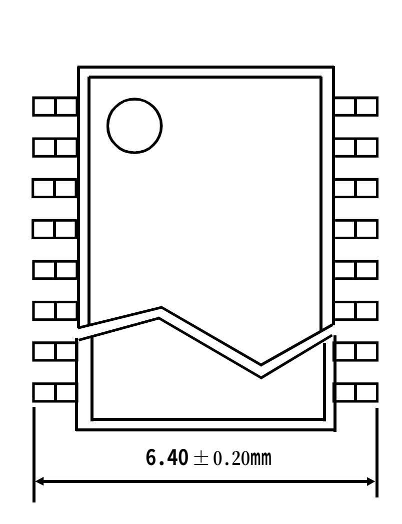
Plastic Dual Inline Package (PDIP-20)
Dimensions in Inches and (Millimeters)

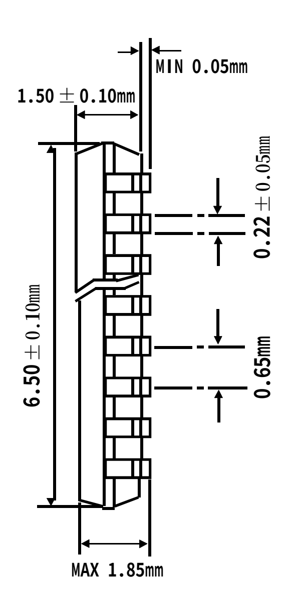


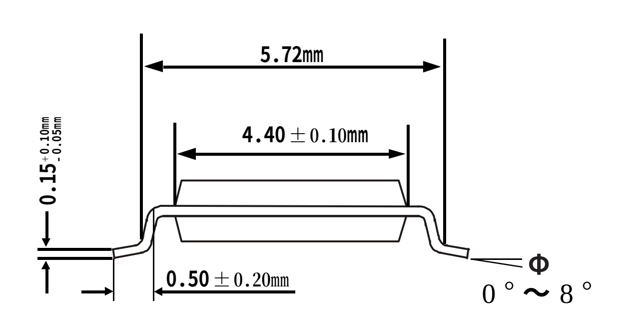
LSSOP-20 封装尺寸图

超小封装 LSSOP-20(仅为 6.4mm x 6.4mm),尺寸只有常规的 SOP-8 大小

PACKAGE：PLASTIC SHRINK SMALL OUTLINE (LSSOP-20, 6.4mm × 6.4mm)

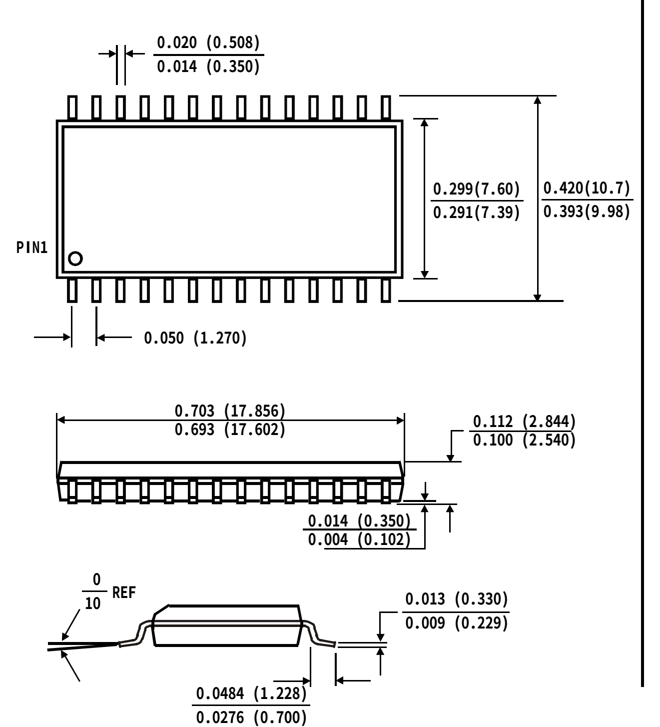






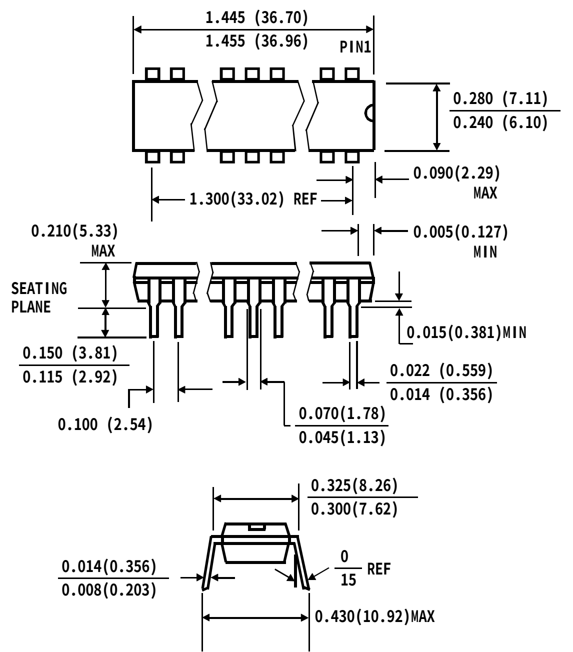
SOP-28 封装尺寸图

28-PIN SMALL OUTLINE PACKAGE (SOP-28)

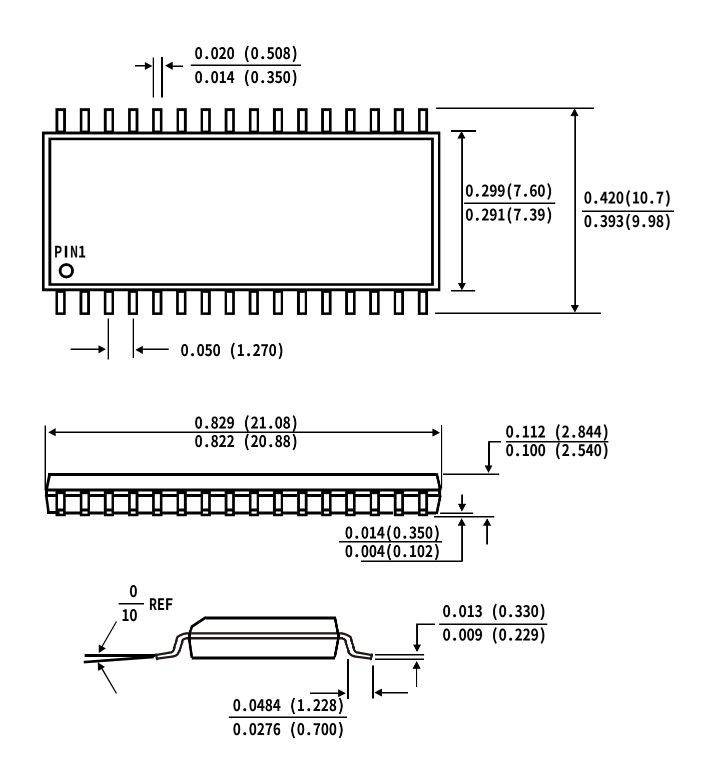


SKDIP-28 封装尺寸图

28-PIN PLASTIC DUAL-IN-LINE PACKAGE (SKDIP-28)

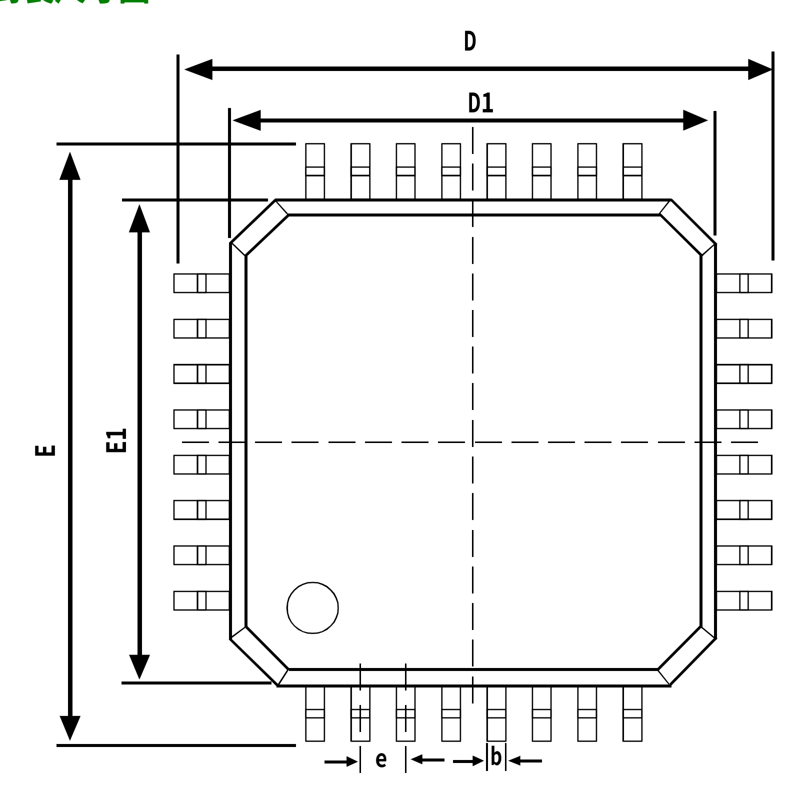


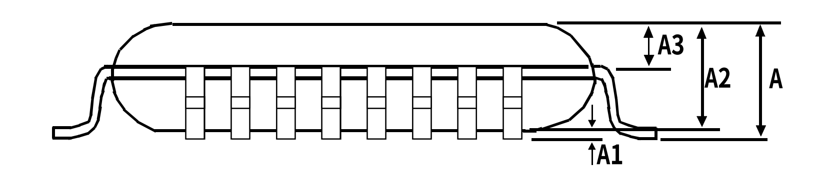
SOP-32 封装尺寸图

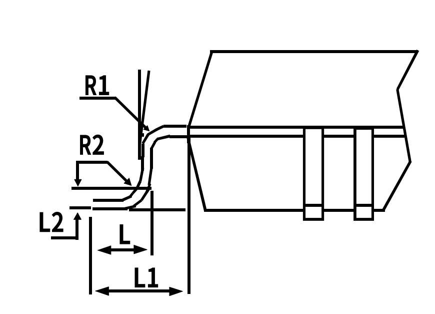


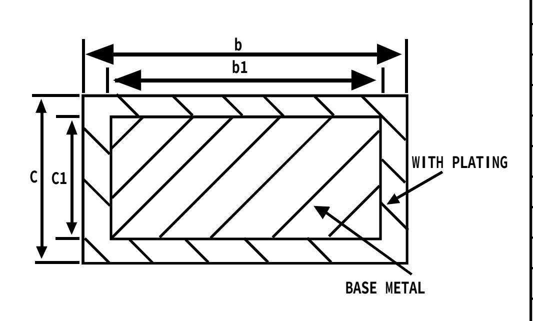
32-PIN SMALL OUTLINE PACKAGE (SOP-32)
0.020 (0.508)
0.014 (0.350)
0.299(7.60)
0.291(7.39)
0.420(10.7)
0.393(9.98)
PIN1
0.050 (1.270)
0.829 (21.08)
0.822 (20.88)
0.112 (2.844)
0.100 (2.540)
0.014(0.350)
0.004(0.102)
0
10
REF
0.013 (0.330)
0.009 (0.229)
0.0484 (1.228)
0.0276 (0.700)

LQFP-32 封装尺寸图







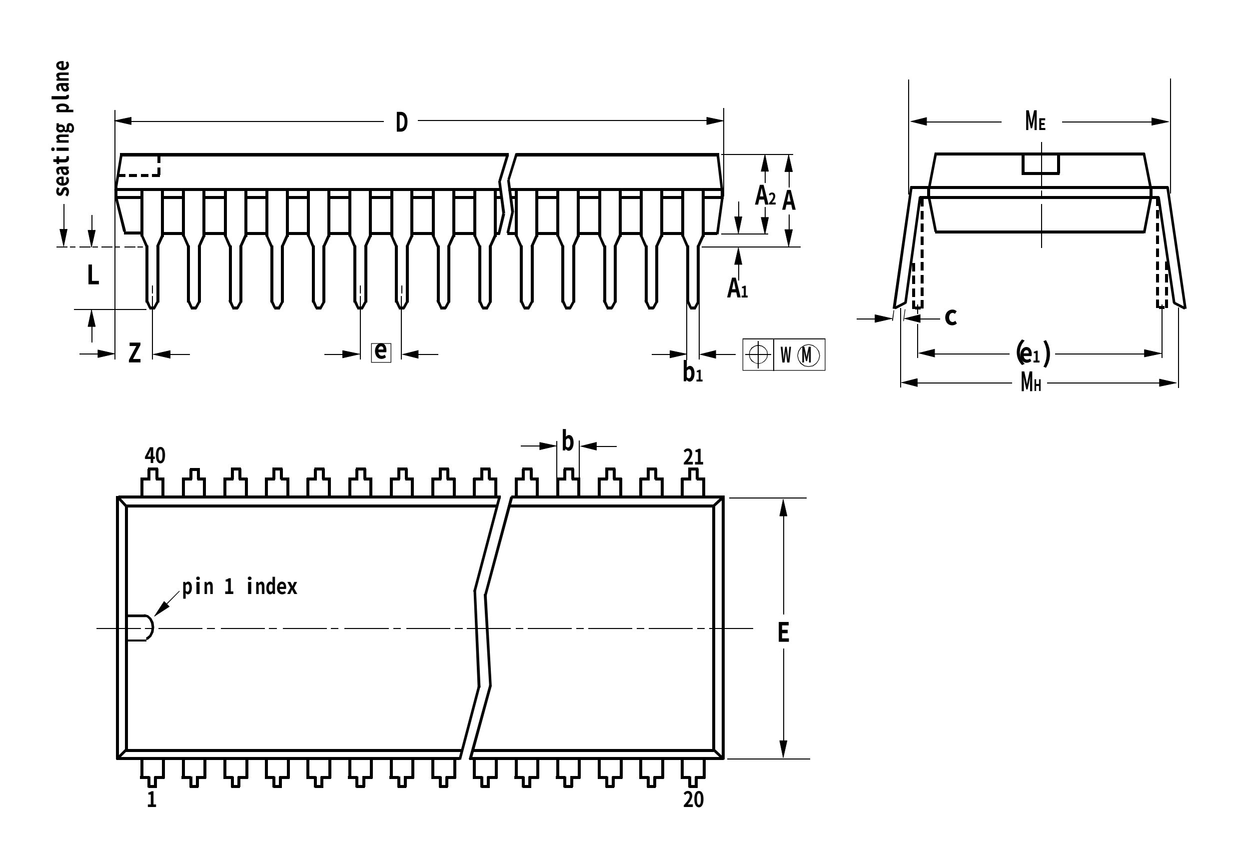


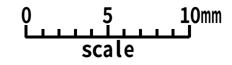
COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.32	-	0.43
b1	0.31	0.35	0.39
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.80BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-
Φ	0°	3.5°	7°
Φ1	0°	–	–
Φ2	11°	12°	13°
Φ3	11°	12°	13°

2.4.2 STC12C5A60AD/S2系列单片机封装尺寸图

PDIP-40 封装尺寸图

PDIP40: plastic dual in-line package;40 leads(600 mil)





DIMENSIONS(inch dimensions are derived from the original mm dimensions)

UNIT	A max.	A_1 min.	A_2 max.	b	b_1	c	D[1]	E[1]	e	e_1	L	M_E	M_H	w	Z[1] max.
mm	4.7	0.51	4.0	1.70 1.14	0.53 0.38	0.36 0.23	52.5 51.5	14.1 13.7	2.54	15.24	3.60 3.05	15.8 15.24	17.42 15.90	0.254	2.25
inches	0.19	0.020	0.16	0.067 0.045	0.021 0.015	0.014 0.009	2.067 2.028	0.56 0.54	0.10	0.60	0.14 0.12	0.62 0.60	0.69 0.63	0.01	0.089

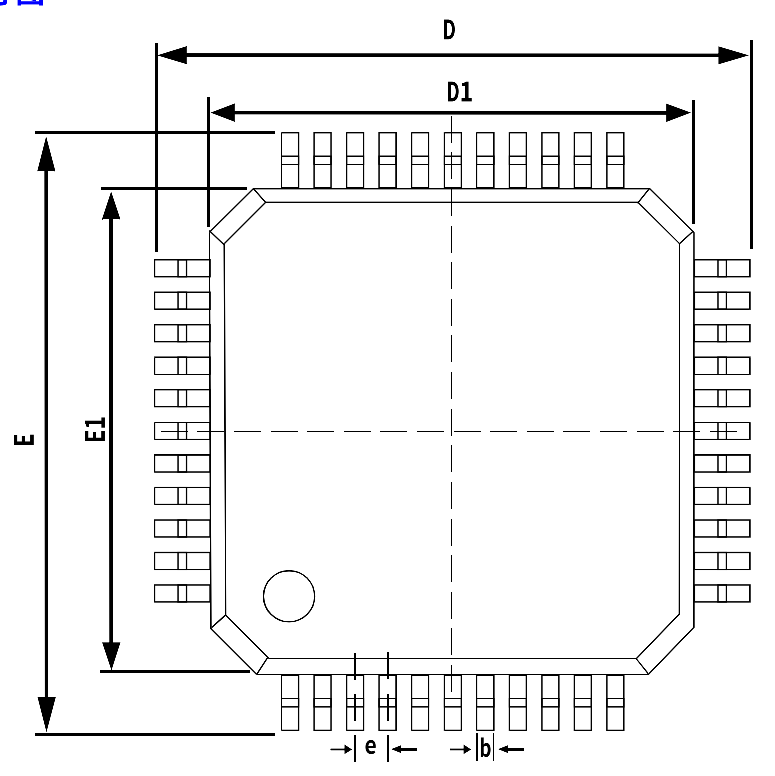
Note

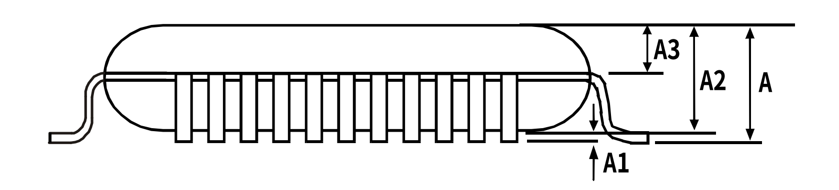
1.Plastic or metal protrusion of 0.25 mm maximum per side are not included

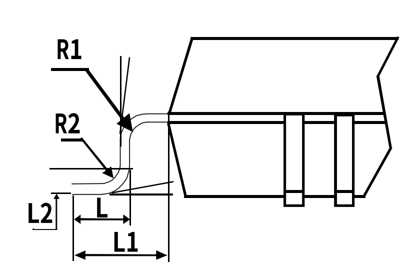
OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT129-1	051G08	MO-015	SC-511-40			~~95-01-14~~ 99-12-27

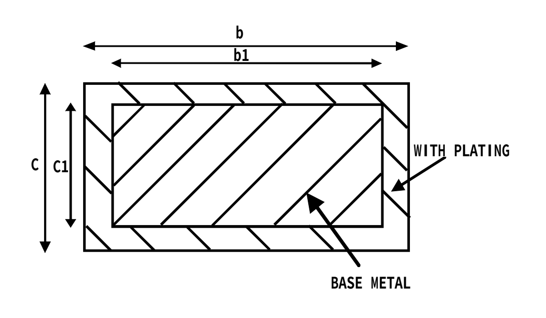
LQFP-44 封装尺寸图

LQFP-44 OUTLINE PACKAGE





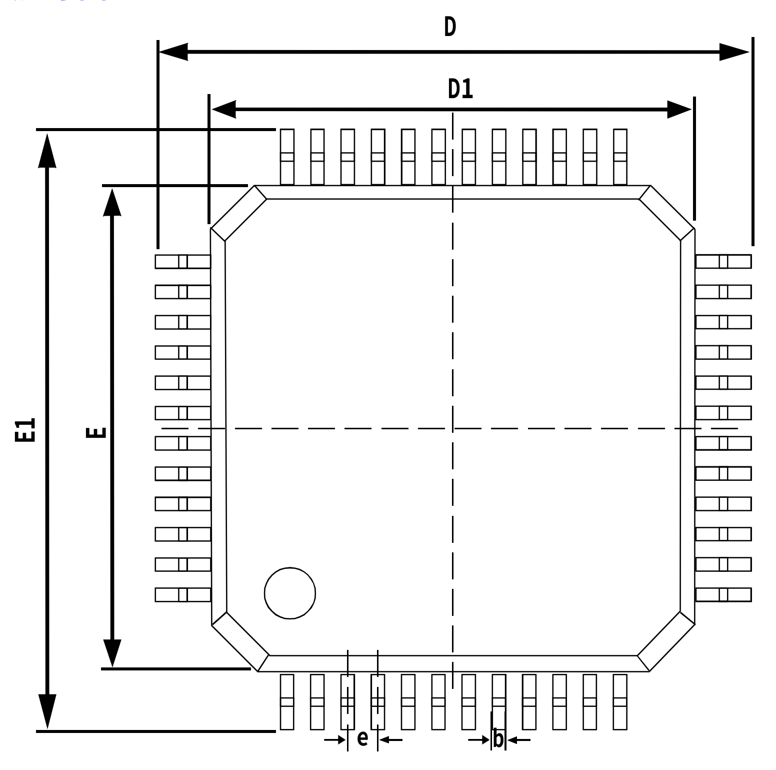


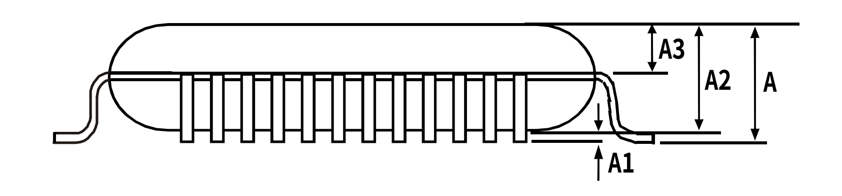


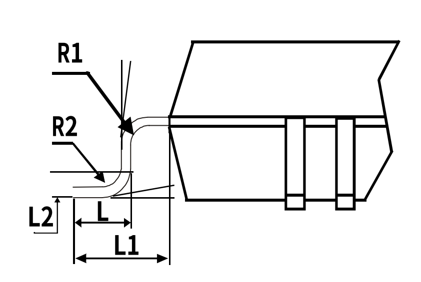
COMMON DIMENSIONS (UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	-	-	1.20
A1	0.05	-	0.15
A2	0.95	1.00	1.05
A3	0.39	0.44	0.49
b	0.31	-	0.44
b1	0.30	0.35	0.40
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
e	0.80BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-
Φ	0°	3.5°	7°
Φ1	0°	–	–
Φ2	11°	12°	13°
Φ3	11°	12°	13°

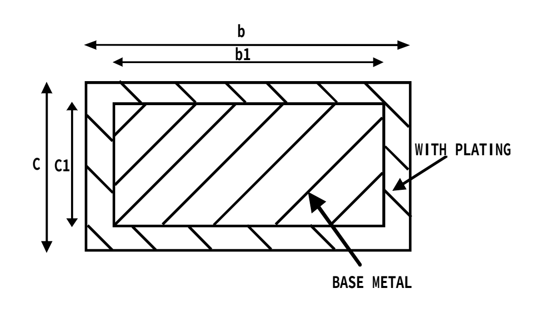
LQFP-48 封装尺寸图

LQFP-48 OUTLINE PACKAGE





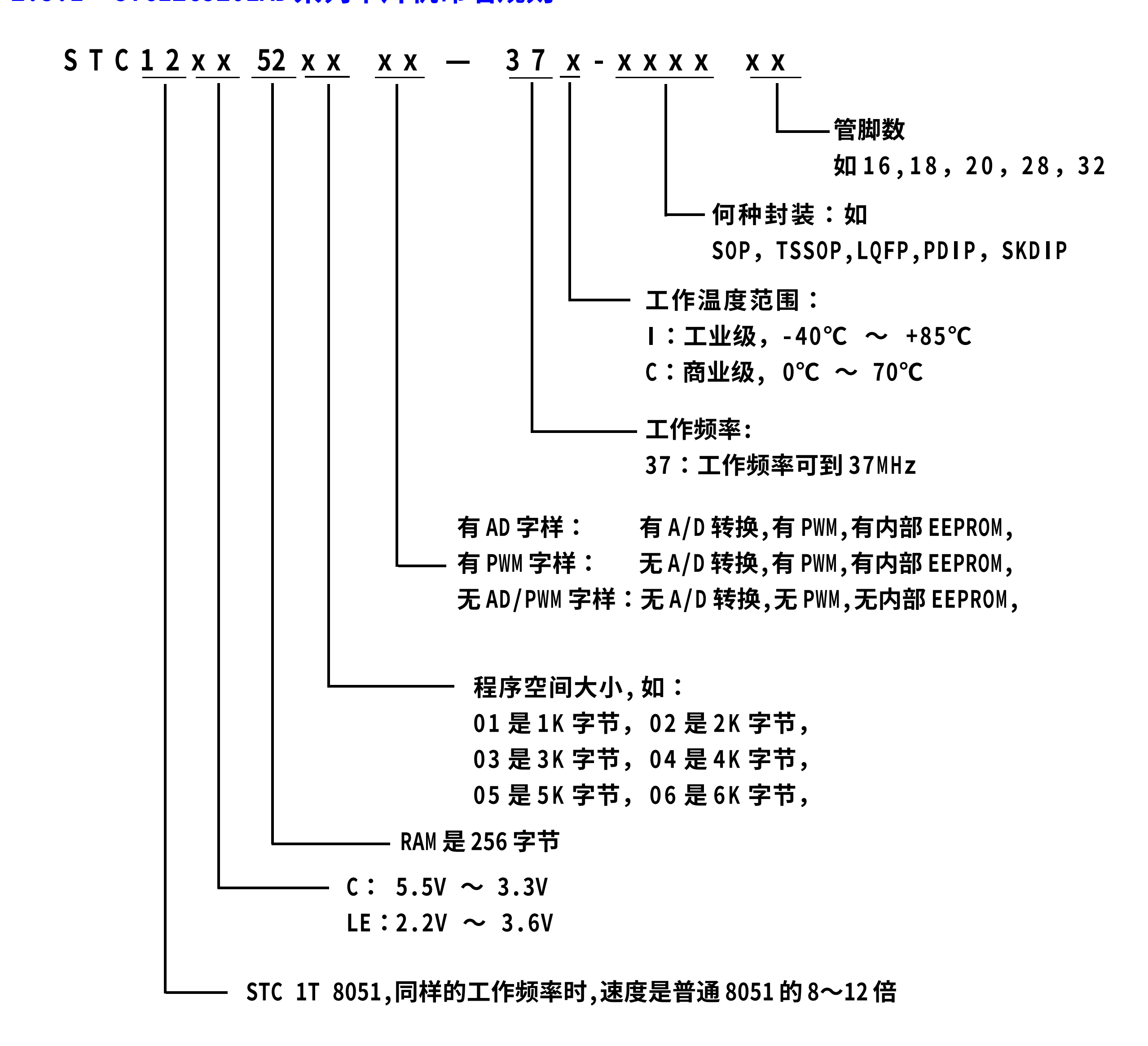




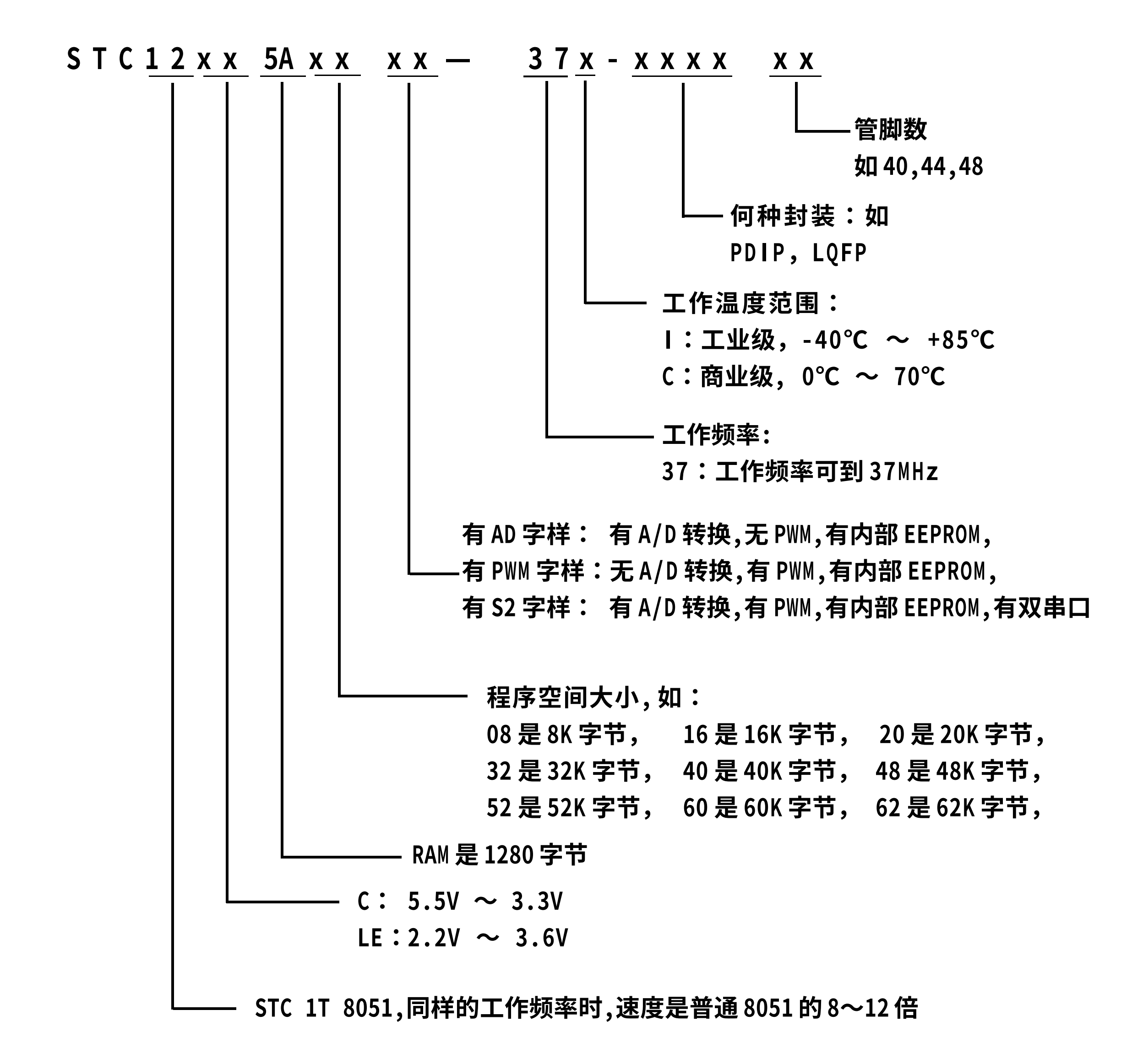
SYMBOL	MIN	NOM	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	-	0.27
b1	0.17	0.20	0.23
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.50BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-

2.5 STC12系列单片机命名规则

2.5.1 STC12C5201AD系列单片机命名规则



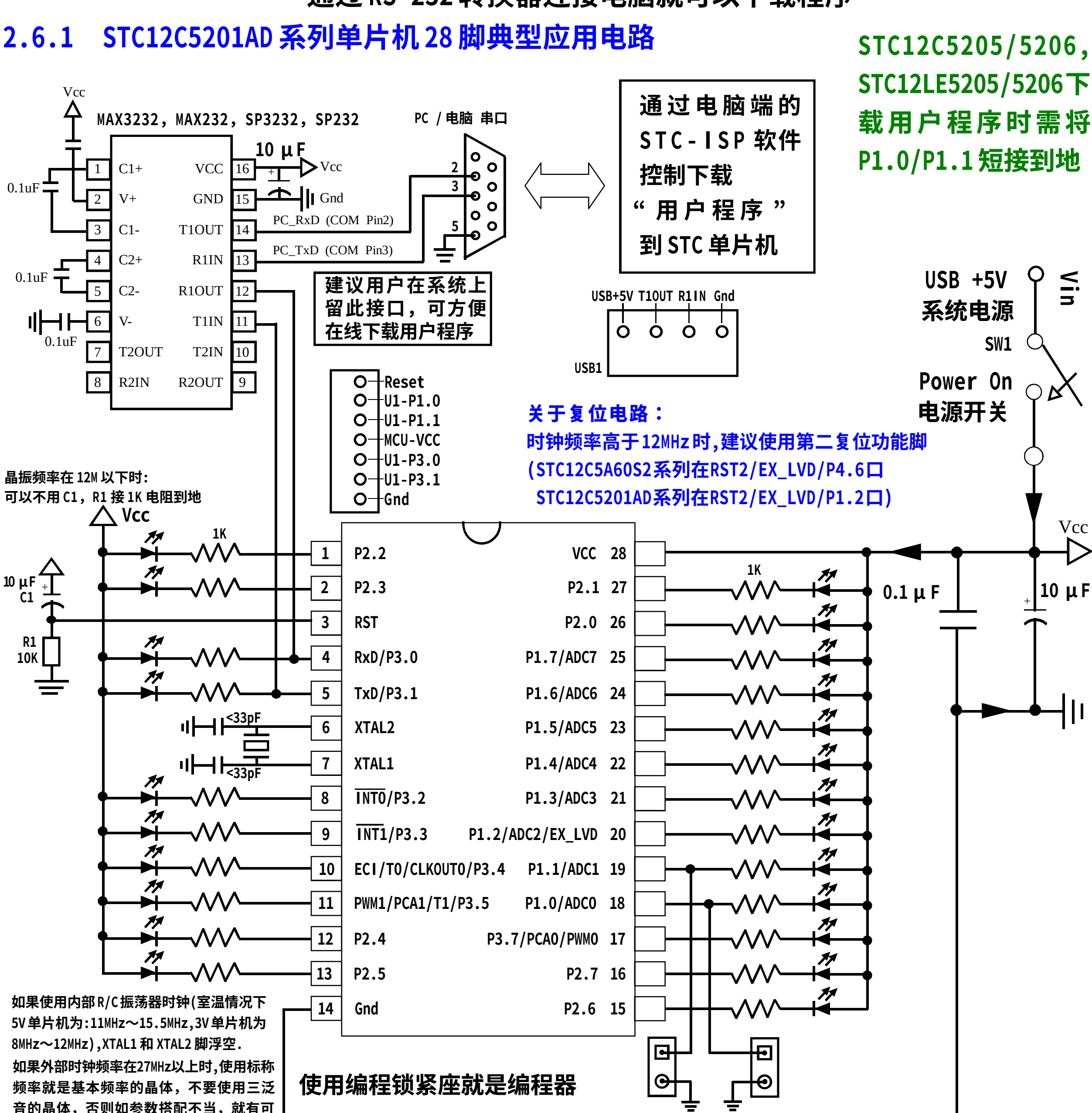
2.5.2 STC12C5A60AD/S2系列单片机命名规则



2.6 STC12C5201AD系列单片机典型应用电路

----通过RS-232转换器连接电脑就可以下载程序

2.6.1 STC12C5201AD 系列单片机 28 脚典型应用电路



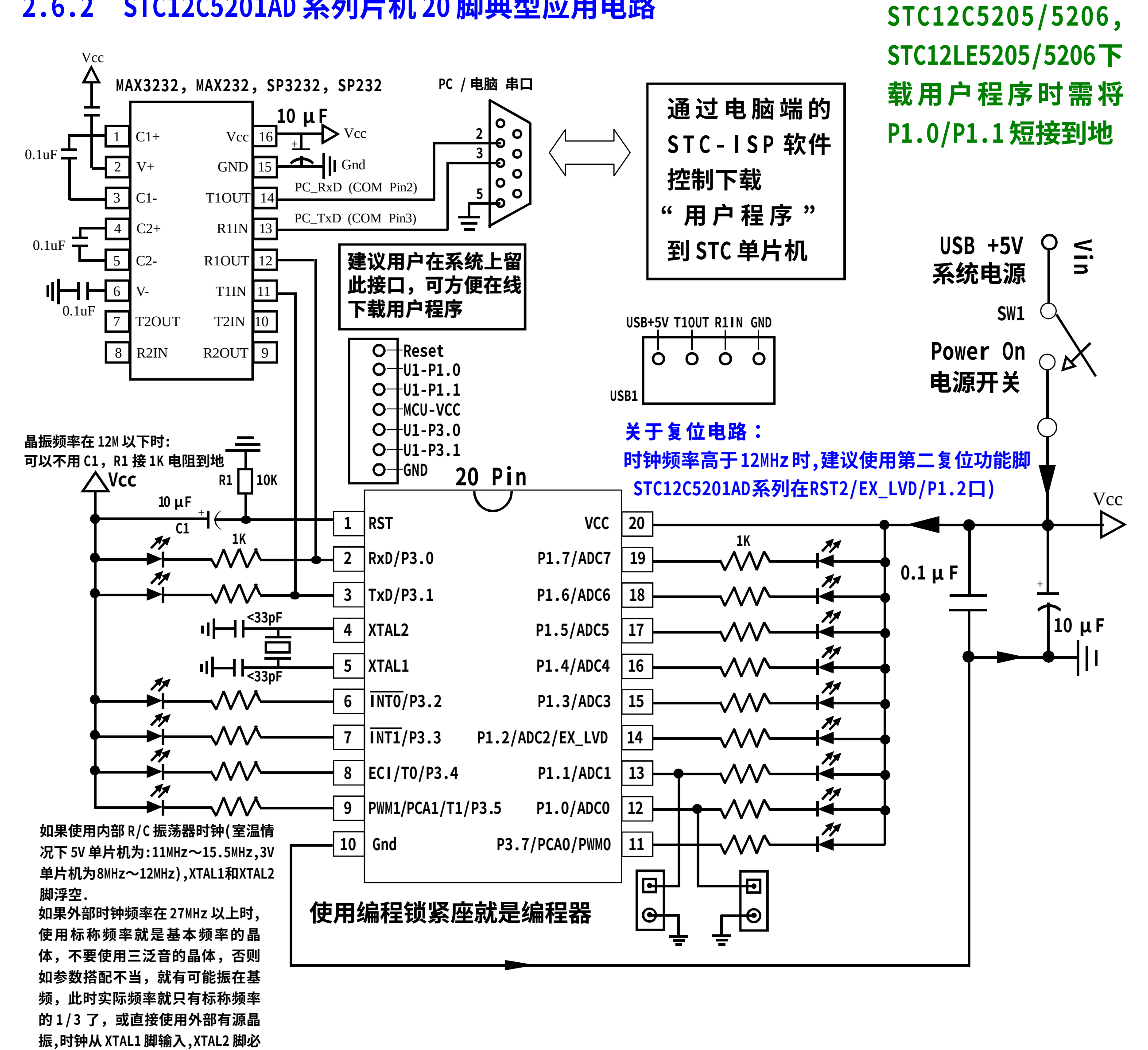
此线路已做成一个STC12C5201AD系列单片机 ISP下载编程工具

用户在自己的目标系统上，如将P3.0/P3.1经过RS-232电平转换器转换后连接到电脑的普通RS-232串口，就可以在系统编程/升级用户软件。建议如果用户板上无RS-232电平转换器，应引出一个插座，含Gnd / P3.1 / P3.0 / Vcc 四个信号线，这样就可以在用户系统上直接编程了。当然如能引出Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为好，因为可以通过P1.0/P1.1禁止ISP下载程序。如果能将Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 / Reset 七个信号线引出就更好了，这样可以很方便的使用“脱机下载板（无需电脑）”。

关于ISP编程的原理及应用指南详见“STC12C5201AD系列单片机开发/编程工具说明”部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的RS-232电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需几秒钟，速度比普通的通用编程器快很多，故无须买第三方的高价编程器。

电脑端STC-ISP软件从网站<u>www.MCU-Memory.com</u>下载

2.6.2 STC12C5201AD 系列片机 20 脚典型应用电路



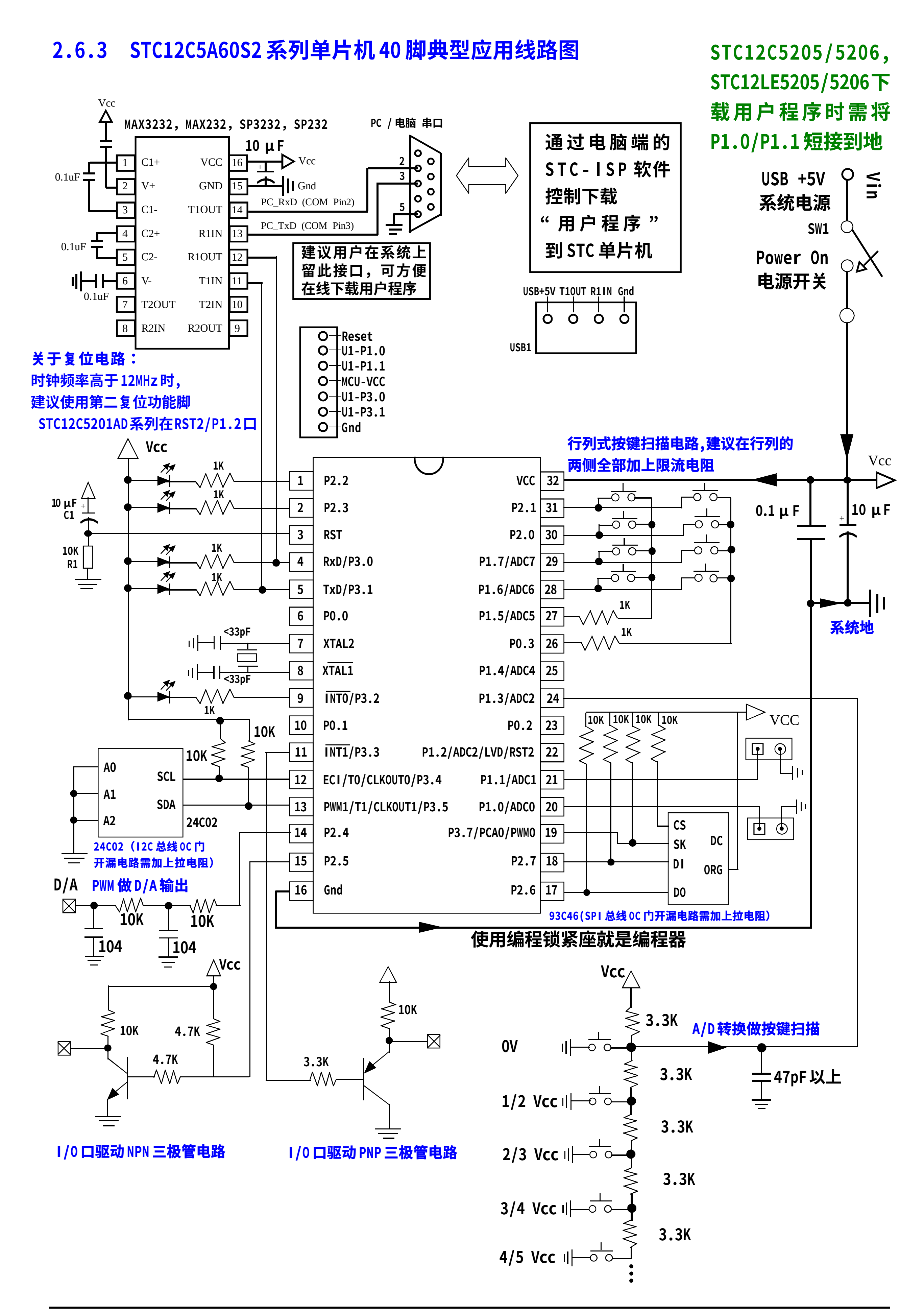
此线路已做成一个 STC12C5201AD 系列单片机 ISP 下载编程工具，可直接赠送给客户

用户在自己的目标系统上，如将 P3.0/P3.1 经过 RS-232 电平转换器转换后连接到电脑的普通 RS-232 串口，就可以在系统编程 / 升级用户软件。建议如果用户板上无 RS-232 电平转换器，应引出一个插座，含 Gnd / P3.1 / P3.0 / Vcc 四个信号线,这样就可以在用户系统上直接编程了。当然如能引出 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为好，因为可以通过 P1.0/P1.1 禁止 ISP 下载程序。如果能将 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 / Reset 七个信号线引出就更好了，这样可以很方便的使用“脱机下载板（无需电脑）”。

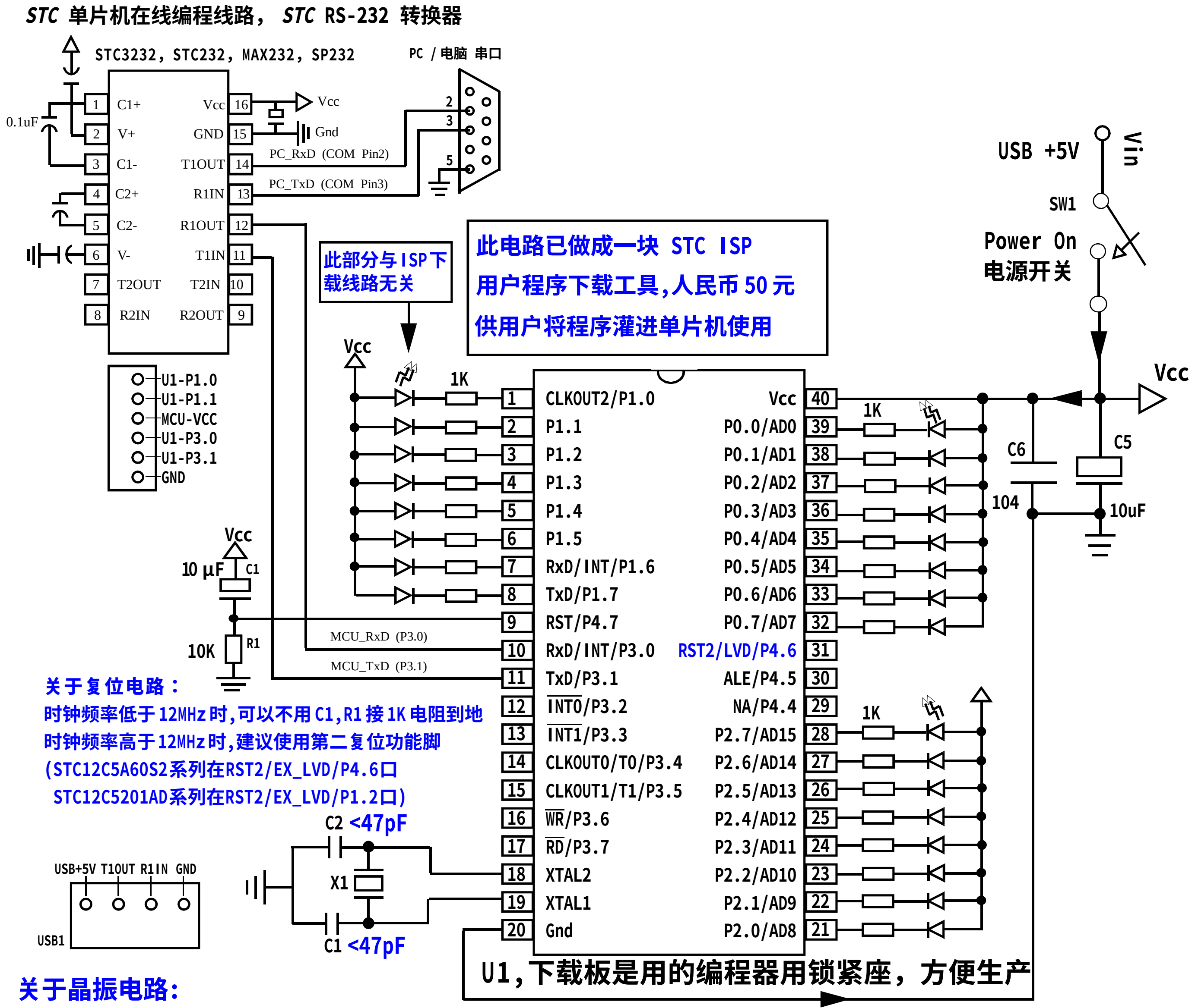
关于 ISP 编程的原理及应用指南详见“STC12C5201AD 系列单片机开发 / 编程工具说明”部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的 RS-232 电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需几秒钟，速度比普通的通用编程器快很多，故无须买第三方的高价编程器。

电脑端 STC-ISP 软件从网站 www.MCU-Memory.com 下载

2.6.3 STC12C5A60S2系列单片机40脚典型应用线路图



2.6.4 STC12C5A60S2系列单片机典型应用电路

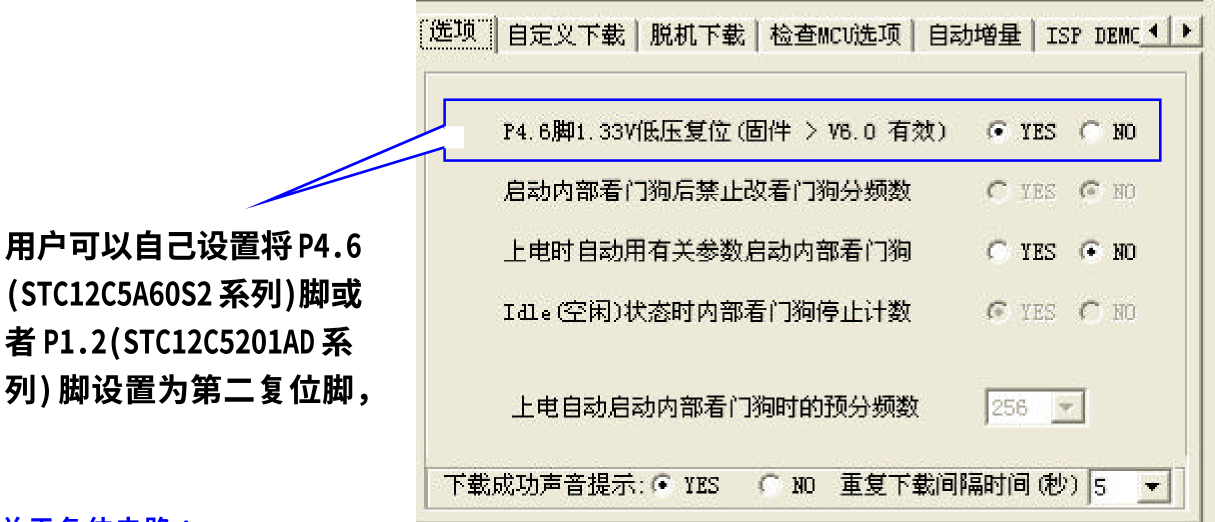


如果外部时钟频率在33MHz以上时,建议直接使用外部有源晶振

如果使用内部R/C振荡器时钟(室温情况下5V单片机为:11MHz～15.5MHz,3V单片机为8MHz～12MHz),XTAL1和XTAL2脚浮空.

如果外部时钟频率在27MHz以上时,使用标称频率就是基本频率的晶体，不要使用三泛音的晶体，否则如参数搭配不当，就有可能振在基频，此时实际频率就只有标称频率的1/3了，或直接使用外部有源晶振,时钟从XTAL1脚输入,XTAL2脚必须浮空.

2.7 新增第二复位功能脚选择与应用

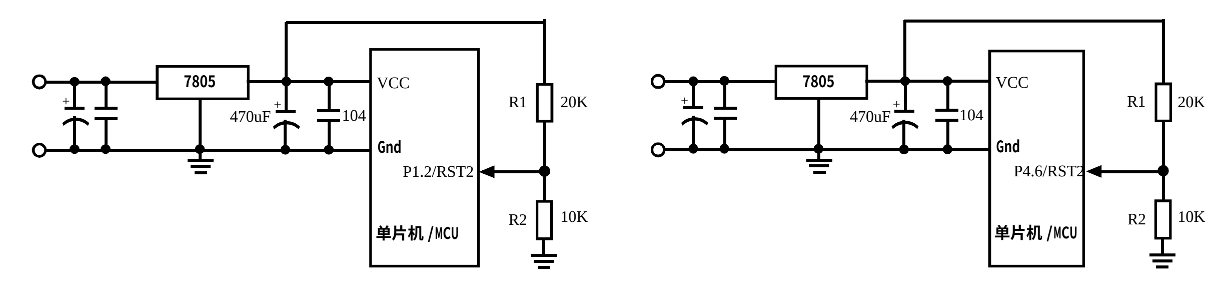


用户可以自己设置将P4.6(STC12C5A60S2系列)脚或者P1.2(STC12C5201AD系列)脚设置为第二复位脚，

关于复位电路：

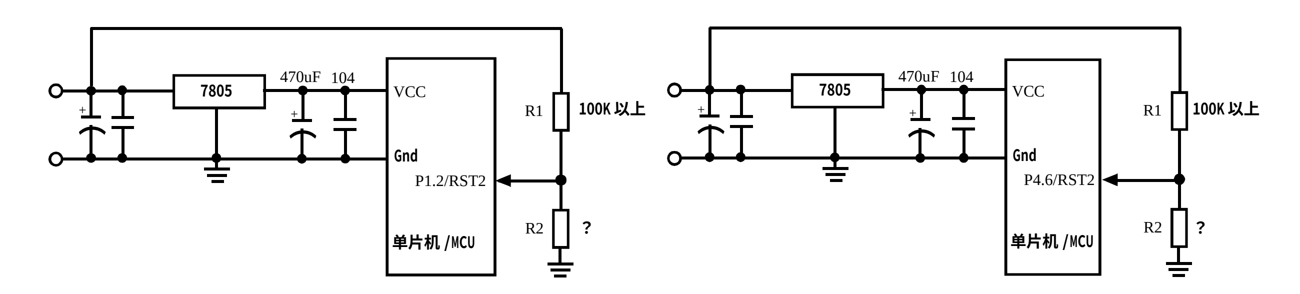
时钟频率高于12MHz时,建议使用第二复位功能脚(STC12C5A60S2系列在RST2/EX_LVD/P4.6口
STC12C5201AD系列在RST2/EX_LVD/P1.2口)

利用增加的外部低压检测LVD功能作外部低压检测复位脚，典型应用线路图



STC12C5201AD系列外部低压检测LVD在P1.2口　　　STC12C5A60S2系列外部低压检测LVD在P4.6口

上图中，稳压块7805后端的直流电是5V,稳压块7805后端的直流电电掉到4V附近时，上图中的电阻R1和R2将4V附近的电压分压到低于低压检测门槛电压(1.33V附近)。此时第二复位功能脚RST2就让CPU处于复位状态,当稳压块7805后端的直流电压高于4V以上时,上图中的电阻R1和R2将4V的电压分压到高于低压检测门槛电压(1.33V附近),单片机就解除复位状态,恢复到正常工作状态.



如交流电在220V时，稳压块7805前端的直流电是11V,当交流电降到160V时，稳压块7805前端的直流电是8.5V，上图中的电阻R1和R2将8.5V的电压分压到低于低压检测门槛电压(1.33V附近)。此时第二复位功能脚RST2就让CPU处于复位状态,当稳压块7805前端的直流电压高于8.5V以上时,上图中的电阻R1和R2将8.5V的电压分压到高于低压检测门槛电压(1.33V附近),单片机就解除复位状态,恢复到正常工作状态.

2.8 指令系统分类总结及与普通8051指令执行时间对比

---与8051指令代码完全兼容，但执行的时间效率大幅提升

---其中 INC DPTR 指令的执行速度大幅提升24倍

---共有12条指令，一个时钟就可以执行完成，平均速度快8～12倍

如果按功能分类，STC12C5A60S2及STC12C5201AD系列单片机指令系统可分为：

1.数据传送类指令；

2.算术操作类指令；

3.逻辑操作类指令；

4.控制转移类指令；

5.布尔变量操作类指令。

按功能分类的指令系统表如下表所示。

传统 12T 的8051 指令执行所需时钟

STC12C5A60S2 系列 指令执行所需时钟

数据传送类指令

助记符		功能说明	字节数	12时钟/机器周期 所需时钟	1时钟/机器周期 所需时钟	效率提升
MOV	A, Rn	寄存器内容送入累加器	1	12	1	12倍
MOV	A, direct	直接地址单元中的数据送入累加器	2	12	2	6倍
MOV	A, @Ri	间接RAM中的数据送入累加器	1	12	2	6倍
MOV	A, #data	立即送入累加器	2	12	2	6倍
MOV	Rn, A	累加器内容送入寄存器	1	12	2	6倍
MOV	Rn, direct	直接地址单元中的数据送入寄存器	2	24	4	6倍
MOV	Rn, #data	立即数送入寄存器	2	12	2	6倍
MOV	direct, A	累加器内容送入直接地址单元	2	12	3	4倍
MOV	direct, Rn	寄存器内容送入直接地址单元	2	24	3	8倍
MOV	direct, direct	直接地址单元中的数据送入另一个直接地址单元	3	24	4	6倍
MOV	direct, @Ri	间接RAM中的数据送入直接地址单元	2	24	4	6倍
MOV	direct, #data	立即数送入直接地址单元	3	24	3	8倍
MOV	@Ri, A	累加器内容送间接RAM单元	1	12	3	4倍
MOV	@Ri, direct	直接地址单元数据送入间接RAM单元	2	24	3	8倍
MOV	@Ri, #data	立即数送入间接RAM单元	2	12	3	4倍
MOV	DPTR, #data16	16位立即数送入地址寄存器	3	24	3	8倍
MOVC	A,@A+DPTR	以DPTR为基地址变址寻址单元中的数据送入累加器	1	24	4	6倍
MOVC	A,@A+PC	以PC为基地址变址寻址单元中的数据送入累加器	1	24	4	6倍
MOVX	A,@Ri	逻辑上在外部的片内扩展RAM，(8位地址）送入累加器	1	24	4	6倍
MOVX	A,@DPTR	逻辑上在外部的片内扩展RAM，(16位地址）送入累加器	1	24	3	8倍
MOVX	@Ri,A	累加器送逻辑上在外部的片内扩展RAM（8位地址）	1	24	3	8倍
MOVX	@DPTR ,A	累加器送逻辑上在外部的片内扩展RAM（16位地址）	1	24	3	8倍
MOVX	A,@Ri	物理上在外部的片外扩展RAM，(8位地址）送入累加器	1	24	7	*Note1
MOVX	A,@DPTR	物理上在外部的片外扩展RAM，(16位地址）送入累加器	1	24	7	*Note1
MOVX	@Ri,A	累加器送物理上在外部的片外扩展RAM，（8位地址）	1	24	7	*Note1
MOVX	@DPTR ,A	累加器送物理上在外部的片外扩展RAM，（16位地址）	1	24	7	*Note1
PUSH	direct	直接地址单元中的数据压入堆栈	2	24	4	6倍
POP	direct	出栈送直接地址单元	2	24	3	8倍

Note1: 访问物理上在片外的扩展RAM所需时钟：7 + 2 x ALE_Bus_Speed + RW_Bus_Speed

其中ALE_Bus_Speed由BUS_SPEED控制寄存器中的ALES1/ALES0决定

其中RW_Bus_Speed由BUS_SPEED控制寄存器中的RWS2/RWS1/RWS0决定

算术操作类指令

助记符	功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ADD A, Rn	寄存器内容加到累加器	1	12	2	6倍
ADD A, direct	直接地址单元中的数据加到累加器	2	12	3	4倍
ADD A, @Ri	间接RAM中的数据加到累加器	1	12	3	4倍
ADD A, #data	立即加到累加器	2	12	2	6倍
ADDC A, Rn	寄存器内容带进位加到累加器	1	12	2	6倍
ADDC A, direct	直接地址单元的内容带进位加到累加器	2	12	3	4倍
ADDC A, @Ri	间接RAM内容带进位加到累加器	1	12	3	4倍
ADDC A, #data	立即数带进位加到累加器	2	12	2	6倍
SUBB A, Rn	累加器带借位减寄存器内容	1	12	2	6倍
SUBB A, direct	累加器带借位减直接地址单元的内容	2	12	3	4倍
SUBB A, @Ri	累加器带借位减间接RAM中的内容	1	12	3	4倍
SUBB A, #data	累加器带借位减立即数	2	12	2	6倍
INC A	累加器加1	1	12	2	6倍
INC Rn	寄存器加1	1	12	3	4倍
INC direct	直接地址单元加1	2	12	4	3倍
INC @Ri	间接RAM单元加1	1	12	4	3倍
DEC A	累加器减1	1	12	2	6倍
DEC Rn	寄存器减1	1	12	3	4倍
DEC direct	直接地址单元减1	2	12	4	3倍
DEC @Ri	间接RAM单元减1	1	12	4	3倍
INC DPTR	地址寄存器DPTR加1	1	24	1	24倍
MUL AB	A乘以B	1	48	4	12倍
DIV AB	A除以B	1	48	5	9.6倍
DA A	累加器十进制调整	1	12	4	3倍

逻辑操作类指令

助记符	功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ANL A, Rn	累加器与寄存器相“与”	1	12	2	6倍
ANL A, direct	累加器与直接地址单元相“与”	2	12	3	4倍
ANL A, @Ri	累加器与间接RAM单元相“与”	1	12	3	4倍
ANL A, #data	累加器与立即数相“与”	2	12	2	6倍
ANL direct, A	直接地址单元与累加器相“与”	2	12	4	3倍
ANL direct, #data	直接地址单元与立即数相“与”	3	24	4	6倍
ORL A, Rn	累加器与寄存器相“或”	1	12	2	6倍
ORL A, direct	累加器与直接地址单元相“或”	2	12	3	4倍
ORL A, @Ri	累加器与间接RAM单元相“或”	1	12	3	4倍
ORL A, #data	累加器与立即数相“或”	2	12	2	6倍
ORL direct, A	直接地址单元与累加器相“或”	2	12	4	3倍
ORL direct, #data	直接地址单元与立即数相“或”	3	24	4	6倍
XRL A, Rn	累加器与寄存器相“异或”	1	12	2	6倍
XRL A, direct	累加器与直接地址单元相“异或”	2	12	3	4倍
XRL A, @Ri	累加器与间接RAM单元相“异或”	1	12	3	4倍
XRL A, #data	累加器与立即数相“异或”	2	12	2	6倍
XRL direct, A	直接地址单元与累加器相“异或”	2	12	4	3倍
XRL direct, #data	直接地址单元与立即数相“异或”	3	24	4	6倍
CLR A	累加器清“0”	1	12	1	12倍
CPL A	累加器求反	1	12	2	6倍
RL A	累加器循环左移	1	12	1	12倍
RLC A	累加器带进位位循环左移	1	12	1	12倍
RR A	累加器循环右移	1	12	1	12倍
RRC A	累加器带进位位循环右移	1	12	1	12倍
SWAP A	累加器半字节交换	1	12	1	12倍

控制转移类指令

助记符	功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ACALL addr11	绝对（短）调用子程序	2	24	6	4倍
LCALL addr16	长调用子程序	3	24	6	4倍
RET	子程序返回	1	24	4	6倍
RETI	中断返回	1	24	4	6倍
AJMP addr11	绝对（短）转移	2	24	3	8倍
LJMP addr16	长转移	3	24	4	6倍
SJMP rel	相对转移	2	24	3	8倍
JMP @A+DPTR	相对于DPTR的间接转移	1	24	3	8倍
JZ rel	累加器为零转移	2	24	3	8倍
JNZ rel	累加器非零转移	2	24	3	8倍
CJNE A, direct, rel	累加器与直接地址单元比较，不相等则转移	3	24	5	4.8倍
CJNE A, #data, rel	累加器与立即数比较，不相等则转移	3	24	4	6倍
CJNE Rn, #data, rel	寄存器与立即数比较，不相等则转移	3	24	4	6倍
CJNE @Ri, #data, rel	间接RAM单元与立即数比较，不相等则转移	3	24	5	4.8倍
DJNZ Rn, rel	寄存器减1，非零转移	3	24	4	6倍
DJNZ direct, rel	直接地址单元减1，非零转移	3	24	5	4.8倍
NOP	空操作	1	12	1	12倍

布尔变量操作类指令

助记符	功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
CLR C	清0进位位	1	12	1	12倍
CLR bit	清0直接地址位	2	12	4	3倍
SETB C	置1进位位	1	12	1	12倍
SETB bit	置1直接地址位	2	12	4	3倍
CPL C	进位位求反	1	12	1	12倍
CPL bit	直接地址位求反	2	12	4	3倍
ANL C, bit	进位位和直接地址位相“与”	2	24	3	8倍
ANL C, $\overline{bit}$	进位位和直接地址位的反码相“与”	2	24	3	8倍
ORL C, bit	进位位和直接地址位相“或”	2	24	3	8倍
ORL C, $\overline{bit}$	进位位和直接地址位的反码相“或”	2	24	3	8倍
MOV C, bit	直接地址位送入进位位	2	12	3	4倍
MOV bit, C	进位位送入直接地址位	2	24	3	8倍
JC rel	进位位为1则转移	2	24	3	8倍
JNC rel	进位位为0则转移	2	24	3	8倍
JB bit, rel	直接地址位为1则转移	3	24	4	6倍
JNB bit, rel	直接地址位为0则转移	3	24	4	6倍
JBC bit, rel	直接地址位为1则转移，该位清0	3	24	5	4.8倍

指令执行速度效率提升总结：

指令系统共包括 111 条指令，其中：

执行速度快 24 倍的	共 1 条
执行速度快 12 倍的	共 12 条
执行速度快 9.6 倍的	共 1 条
执行速度快 8 倍的	共 20 条
执行速度快 6 倍的	共 38 条
执行速度快 4.8 倍的	共 4 条
执行速度快 4 倍的	共 21 条
执行速度快 3 倍的	共 14 条

根据对指令的使用频率分析统计，STC12 系列 1T 的 8051 单片机比普通的 8051 单片机在同样的工作频率下运行速度提升了 8～12 倍。

指令执行时钟数统计（供参考）：

指令系统共包括 111 条指令，其中：

1 个时钟就可执行完成的指令	共 12 条
2 个时钟就可执行完成的指令	共 20 条
3 个时钟就可执行完成的指令	共 39 条
4 个时钟就可执行完成的指令	共 33 条
5 个时钟就可执行完成的指令	共 5 条
6 个时钟就可执行完成的指令	共 2 条

2.9 特殊功能寄存器映像 SFR Mapping

	Bit Addressable 可位操作	Non Bit Addressable 不可以位操作(寄存器地址不可以被8整除的不可以进行位操作)							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h		CH 0000,0000	CCAP0H 0000,0000	CCAP1H 0000,0000					FFh
F0h	B 0000,0000		PCA_PWM0 xxxx,xx00	PCA_PWM1 xxxx,xx00					F7h
E8h		CL 0000,0000	CCAP0L 0000,0000	CCAP1L 0000,0000					EFh
E0h	ACC 0000,0000								E7h
D8h	CCON 00xx,xx00	CMOD 0xxx,0000	CCAPM0 x000,0000	CCAPM1 x000,0000					DFh
D0h	PSW 0000,0000								D7h
C8h	P5 xxxx,1111	P5M1 xxxx,0000	P5M0 xxxx,0000			SPSTAT 00xx,xxxx	SPCTL 0000,0100	SPDAT 0000,0000	CFh
C0h	P4 1111,1111	WDT_CONTR xx00,0000	IAP_DATA 1111,1111	IAP_ADDRH 0000,0000	IAP_ADDRL 0000,0000	IAP_CMD xxxx,xx00	IAP_TRIG xxxx,xxxx	IAP_CONTR 0000,1000	C7h
B8h	IP 0000,0000	SADEN		P4SW x000,xxxx	ADC_CONTR 0000,0000	ADC_RES 0000,0000	ADC_RESL 0000,0000		BFh
B0h	P3 1x11,1111	P3M1 0000,0000	P3M0 0000,0000	P4M1 0000,0000	P4M0 0000,0000	IP2 xxxx,xx00	IP2H xxxx,xx00	IPH 0000,0000	B7h
A8h	IE 0000,0000	SADDR						IE2 xxxx,xx00	AFh
A0h	P2 1111,1111	BUS_SPEED xx10,x011	AUXR1 0000,0000					TEST_WDT don't use	A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx	S2CON 0000,0000	S2BUF xxxx,xxxx	BRT 0000,0000	P1ASF 0000,0000			9Fh
90h	P1 1111,1111	P1M1 0000,0000	P1M0 0000,0000	P0M1 0000,0000	P0M0 0000,0000	P2M1 0000,0000	P2M0 0000,0000	CLK_DIV xxxx,x000	97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR 0000,0000	WAKE_CLKO 0000,0x00	8Fh
80h	P0 xxxx,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000				PCON 0011,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

注意：寄存器地址能够被 8 整除的才可以进行位操作，不能够被 8 整除的不可以进行位操作

特别标出部分为在 Intel 8052 基础上新增加的特殊功能寄存器,一般用户可不管

新增特殊功能寄存器如何声明地址，举例如下：

汇编语言(新增 P4 口地址声明) ： P4 EQU 0C0H

C 语言 (新增 P4 口地址声明) ： sfr P4 = 0xC0

```
sbit  P40 = 0xC0;
sbit  P41 = 0xC1;
sbit  P42 = 0xC2;
```

STC12C5201AD 系列 8051 单片机内核特殊功能寄存器 C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ACC	E0h	Accumulator									0000,0000
B	F0h	B Register									0000,0000
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P	0000,0000
SP	81h	Stack Pointer									0000,0111
DPL	82h	Data Pointer Low Byte									0000,0000
DPH	83h	Data Pointer High Byte									0000,0000

STC12C5201AD 系列 8051 单片机系统管理特殊功能寄存器 System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
CLK_DIV	97h	Clock Divder	-	-	-	-	-	CLKS2	CLKS1	CLKS0	xxxx,x000

STC12C5201AD 系列 8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	-	-	-	-	P0.3	P0.2	P0.1	P0.0	xxxx,1111
P0M1	93h										0000,0000
P0M0	94h										0000,0000
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P1M1	91h										0000,0000
P1M0	92h										0000,0000
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P2M1	95h										0000,0000
P2M0	96h										0000,0000
P3	B0h	8-bit Port 3	P3.7	-	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1x11,1111
P3M1	B1h										0000,0000
P3M0	B2h										0000,0000

STC12C5201AD 系列 8051 单片机 定时器 特殊功能寄存器 Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TL0	8Ah	Timer / Counter 0 Low Byte									0000,0000
TH0	8Ch	Timer / Counter 0 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000

STC12C5201AD 系列 8051 单片机 串行口 特殊功能寄存器 Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000

STC12C5201AD 系列 8051 单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	C1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

STC12C5201AD 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
WAKE_CLKO	8Fh	CLK_Output Powerdown_Wakeup Control Register	PCAWAKEUP	RXD_PIN_IE	T1_PIN_IE	T0_PIN_IE	LVD_WAKE	-	T1CLKO	T0CLKO	0000,0x00
ADC_CONTR	BCh	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000,0000
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF	0xxx,0000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000

STC12C5201AD 系列 8051 单片机 PCA/PWM 特殊功能寄存器 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF	0xxx,0000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00

STC12C5201AD 系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IAP_DATA	C2h	ISP/IAP Flash Data Register									1111,1111
IAP_ADDRH	C3h	ISP/IAP Flash Address High									0000,0000
IAP_ADDRL	C4h	ISP/IAP Flash Address Low									0000,0000
IAP_CMD	C5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,x000
IAP_TRIG	C6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
IAP_CONTR	C7h	ISP/IAP Control Register	IAPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0	0000,x000

STC12C5201AD 系列 8051 单片机 时钟输出和掉电唤醒寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WAKE_CLKO	8Fh	Clk_Output Powerdown_Wakeup Control register	PCAWAKEUP	RXD_PIN_IE	T1_PIN_IE	T0_PIN_IE	LVD_WAKE	BRTCLKO	T1CLKO	T0CLKO	0000,0x00

STC12C5A60S2 系列单片机新增加的特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P4	C0h	8 - bit Port 4	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0	1111,1111
P4M1	B3h	P4 Configuration 1									0000,0000
P4M0	B4h	P4 Configuration 0									0000,0000
P4SW	BBh	Port - 4 switch		LVD_P4.6	ALE_P4.5	NA_P4.4					x000,xxxx
P5	C8h	8 - bit Port 5	-	-	-	-	P5.3	P5.2	P5.1	P5.0	xxxx,1111
P5M1	C9h	P5 Configuration 1									0000,0000
P5M0	CAh	P5 Configuration 0									0000,0000
ADC_RESL	BEh	ADC Result low									0000,0000
SPSTAT	CDh	SPI Status register	SPIF	WCOL							00xx,xxxx
SPCTL	CEh	SPI Control register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000.0100
SPDAT	CFh	SPI Data register									0000,0000
AUXR1	A2h	Auxiliary register 1		PCA_P4	SPI_P4	S2_P4	GF2	ADRJ		DPS	0000,0000
IE2	AFh	Interrupt Enable 2							ESPI	ES2	xxxx,xx00
IP2	B5h	2rd Interrupt Priority Low register							PSPI	PS2	xxxx,xx00
IP2H	B6h	2rd Interrupt Priority High register							PSPIH	PS2H	xxxx,xx00
S2CON	9Ah	Serial 2 Control register	S2SM0	S2SM1	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0000,0000
S2BUF	9Bh	Serial 2 Buffer									xxxx,xxxx
BRT	9Ch	Serial 2 Baud-Rate timer									0000,0000
BUS_SPEED	A1h	Bus-Speed Control			ALES1	ALES0		RWS2	RWS1	RWS0	xx10,x011

2.10 中断优先级及中断寄存器

2.10.1 中断优先级

STC12C5201AD 系列 STC12C5A60S2 系列单片机 中断优先级及中断查询次序，与 8051 完全兼容

Interrupt Source 中断源	Vector Address 中断 向量地址	Polling Sequence 中断 查询次序	中断 优先级设置 (IPH,IP)	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求标志位	Interrupt Enable Control Bit 中断允许控制位
/INT0	0003H	0(最优先)	PX0H,PX0	0,0	0,1	1,0	1,1	IE0	EX0 / EA
Timer 0	000BH	1	PT0H,PT0	0,0	0,1	1,0	1,1	TF0	ET0 / EA
/INT1	0013H	2	PX1H,PX1	0,0	0,1	1,0	1,1	IE1	EX1 / EA
Timer 1	001BH	3	PT1H,PT1	0,0	0,1	1,0	1,1	TF1	ET1 / EA
UART	0023H	4	PSH, PS	0,0	0,1	1,0	1,1	RI + TI	ES / EA
ADC	002BH	5	PADCH,PADC	0,0	0,1	1,0	1,1	ADC_FLAG	EADC / EA
LVD	0033H	6	PLVDH,PLVD	0,0	0,1	1,0	1,1	LVDF	ELVD / EA
PCA	003BH	7	PPCAH,PPCA	0,0	0,1	1,0	1,1	CF + CCF0 + CCF1	(ECF+ECCF0+ECCF1)/EA
以上是12C5202AD/12C5202PWM/12C5202系列和12C5A60S2/AD/PWM系列共有的中断									
UART2	0043	8	PS2H,PS2	0,0	0,1	1,0	1,1	S2TI +S2RI	ES2 / EA
UART2是12C5A60S2系列独有的第二个串口中断(12C5A60AD系列/12C5A60PWM系列/12C5202AD系列没有UART2)									
SPI	004B	9	PSPIH,PSPI	0,0	0,1	1,0	1,1	SPIF	ESPI / EA
SPI是12C5A60S2/12C5A60AD/12C5A60PWM系列系列系列独有的SPI中断									

通过设置新增加的特殊功能寄存器 IPH 中的相应位，可将中断优先级设为四级，如果只设置 IP，那么中断优先级就只有两级，与传统 8051 单片机两级中断优先级完全兼容。

如果使用 C 语言编程，中断查询次序号就是中断号，例如：

```
void  Int0_Routine(void) interrupt   0;
void  Timer0_Routine(void) interrupt 1;
void  Int1_Routine(void) interrupt   2;
void  Timer1_Routine(void) interrupt 3;
void  UART_Routine(void) interrupt   4;
void  ADC_Routine(void) interrupt    5;
void  LVD_Routine(void) interrupt    6;
void  PCA_Routine(void) interrupt    7;
void  UART2_Routine(void) interrupt  8;
void  SPI_Routine(void) interrupt    9;
```

STC12C5201AD 系列和 STC12C5A60S2 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
IE2	AFh	Interrupt Enable 2							ESPI	ES2	xxxx,xx00
IP2	B5h	2rd Interrupt Priority Low register							PSPI	PS2	xxxx,xx00
IP2H	B6h	2rd Interrupt Priority High register							PSPIH	PS2H	xxxx,xx00
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
WAKE_CLKO	8Fh	CLK_Output Powerdown_Wakeup Control Register	PCAWAKEUP	RXD_PIN_IE	T1_PIN_IE	T0_PIN_IE	LVD_WAKE	BRTCLKO	T1CLKO	T0CLKO	0000,0x00
ADC_CONTR	BCh	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF	0xxx,0000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000

PCA/PWM 特殊功能寄存器 ，其中部分位与 PCA 中断有关

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF	0xxx,0000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00

*** 以上寄存器中标为红色的部分为 STC12C5A60S2/AD/PWM 系列单片机特有的寄存器或者控制位，STC12C5201AD/PWM 系列没有 ***

2.10.2 几个新增加的中断控制位

如果要允许 A/D 转换中断则需要将相应的控制位置 1:

1、将 EADC 置 1，允许 ADC 中断，这是 ADC 中断的中断控制位。

2、将 EA 置 1，打开单片机总中断控制位，此位不打开，也是无法产生 ADC 中断的

A/D 中断服务程序中要用软件清 A/D 中断请求标志位 ADC_FLAG(也是 A/D 转换结束标志位)。

与 A/D 转换有关的特殊功能寄存器表

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000
ADC_CONTR	BCh	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000,0000
ADC_RES	BDh	A/D 转换结果寄存器	-	-	-	-	-	-	-	-	xxxx,xxxx
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

如果要允许低压中断则需要将相应的控制位置 1:

1、将 ELVD 置 1，允许低压检测中断，这是低压中断的中断控制位。

2、将 EA 置 1，打开单片机总中断控制位，此位不打开，也是无法产生低压检测中断的

3、如要在掉电模式时,允许低压检测中断唤醒 CPU,还要将 WAKE_CLKO 寄存器当中的 LVD_WAKE 位置 1

低压检测中断服务程序中要用软件清低压中断请求标志位 LVDF。

低压检测中断发生的条件是 P1.2 口的输入电压低于：

5V 单片机为 1.32V，但有制造误差 +/-5%

3V 单片机为 1.30V，但有制造误差 +/-3%

上电复位后 LVDF 标志位为 1,要由软件清 0，当 P1.2 口外部输入电压低于检测门槛电压时,LVDF = 1 如果要求在掉电模式下外部低压检测中断继续工作,可将CPU从掉电模式唤醒,应将特殊功能寄存器WAKE_CLKO 中的相应控制位 LVD_WAKE 置 1.如果不需要在掉电模式下外部低压检测中断唤醒,建议将 P1ASF 寄存器中的第二位 P12ASF 置 1,否则在掉电模式时,此低压检测电路会继续有几十 uA 的电流消耗.

与外部低压检测 LVD 有关的特殊功能寄存器表

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
WAKE_CLKO	8Fh	Clk_output Powerdown_Wakeup Control register	PCAWAKEUP	RXD_PIN_IE	T1_PIN_IE	T0_PIN_IE	LVD_WAKE	BRTCLKO	T1CLKO	T0CLKO	0000,0x00
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

如果要允许 PCA 中断则需要将相应的控制位置 1:

1、将 ECF/ECCF0/ECCF1 中断允许位需要置 1 的位置 1，允许 PCA 模块中相应的模块产生中断

2、将 EA 置 1，打开单片机总中断控制位，此位不打开，也是无法产生 PCA 中断的

PCA 中断服务程序中要用软件清相应的 PCA 中断请求标志位 CF/CCF0/CCF1。

2.11 定时器 0/ 定时器 1，UART 串口的速度

STC12C5201/12C5201PWM/12C5201AD 系列单片机的 AUXR 寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000

定时器 0 和定时器 1:
STC12C5201AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，定时器 0 和定时器 1 复位后是传统 8051 的速度，即 12 分频，这是为了兼容传统 8051。但也可不进行 12 分频，实现真正的 1T。

T0x12: 0，定时器 0 是传统 8051 速度，12 分频；1，定时器 0 的速度是传统 8051 的 12 倍，不分频

T1x12: 0，定时器 1 是传统 8051 速度，12 分频；1，定时器 1 的速度是传统 8051 的 12 倍，不分频

如果 UART 串口用定时器 1 做波特率发生器，T1x12 位就可以控制 UART 串口是 12T 还是 1T 了。

UART 串口的模式 0:
STC12C5201AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，UART 串口复位后是兼容传统 8051 的。
UART_M0x6: 0，UART 串口的模式 0 是传统 12T 的 8051 速度，12 分频；
1，UART 串口的模式 0 的速度是传统 12T 的 8051 的 6 倍，2 分频
如果用定时器 T1 做波特率发生器时,UART 串口的速度由 T1 的溢出率决定

STC12C5A60PWM/12C5A60AD/12C5A60S2 系列单片机的 AUXR 寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000

BRTR(S2TR): 0，不允许独立波特率发生器运行
1，允许独立波特率发生器运行
S2SMOD: 0，缺省
1，串口 2 / UART2 的波特率 x 2
BRTx12(S2Tx12): 0，独立波特率发生器每 12 个时钟计数一次
1，独立波特率发生器每 1 个时钟计数一次
EXTRAM: 0，允许使用内部扩展的 1024 字节扩展 RAM
1，禁止使用内部扩展的 1024 字节扩展 RAM
S1BRS: 0，缺省，串口 1 波特率发生器选择定时器 1，S1BRS 是串口 1 波特率发生器选择位
1，独立波特率发生器作为串口 1 的波特率发生器，此时定时器 1 得到释放，
可以作为独立定时器使用

注 意 ：
有串口 2 的单片机，串口 2 永远是使用独立波特率发生器(2)作为波特率发生器,串口 2 不能够选择定时器 1 做波特率发生器，
串口 1 可以选择定时器 1 做波特率发生器，也可以选择独立波特率发生器(2)作为波特率发生器，

2.12 STC12 系列单片机内部 / 外部工作时钟可选

STC12C5201AD 系列是 1T 的 8051 单片机，系统时钟兼容传统 8051。

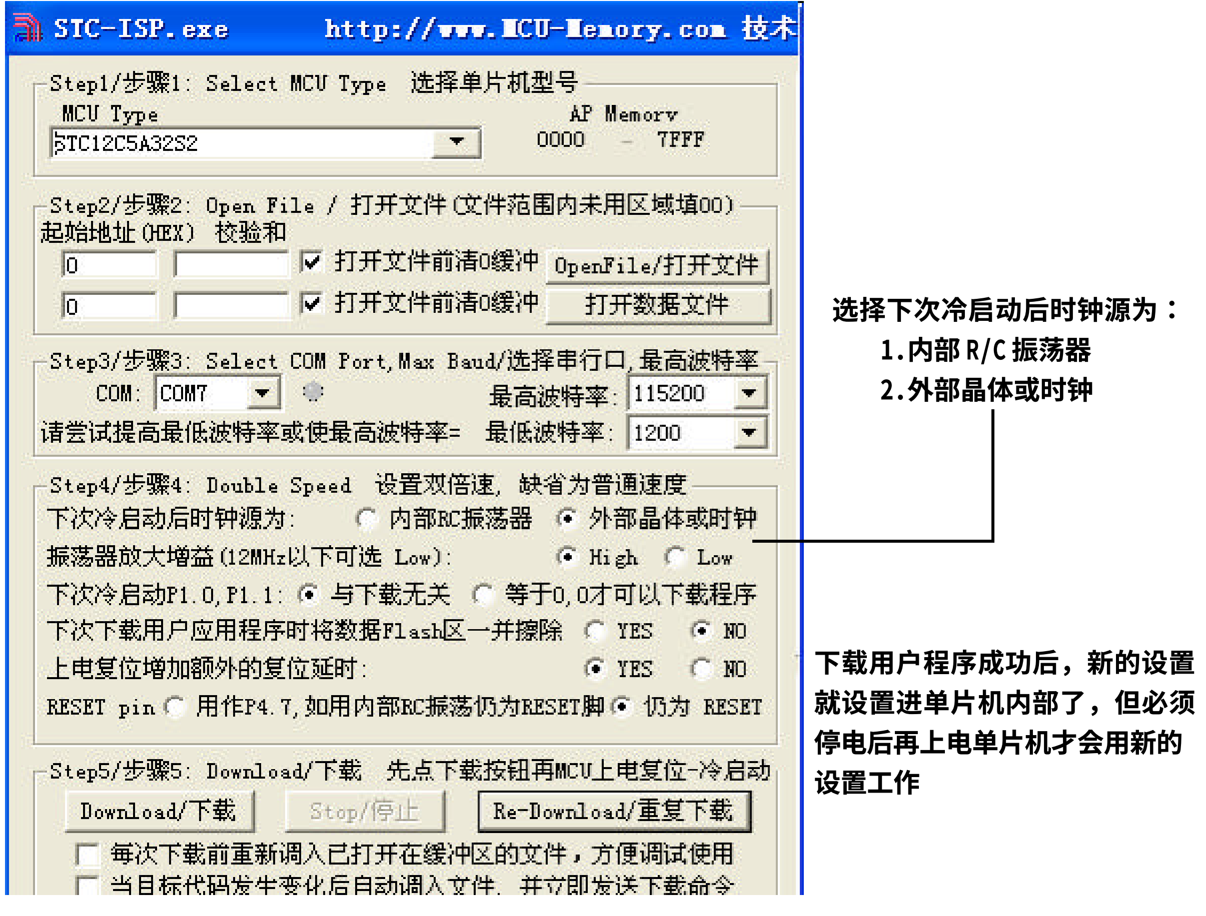
现出厂标准配置是使用芯片内部的 R/C 振荡器，5V 单片机常温下频率是 11MHz - 15.5MHz,3V 单片机常温下频率是 8MHz - 12MHz,因为随着温度的变化，内部 R/C 振荡器的频率会有一些温飘，再加上制造误差，故内部 R/C 振荡器只适用于对时钟频率要求不敏感的场合。

在对 STC12C5201AD 系列单片机进行 ISP 下载用户程序时，可以在选项中选择：

“下次冷启动后时钟源为外部晶体或时钟”

这样下载完用户程序后，停电，再冷启动后单片机的工作时钟使用的就不是内部 R/C 振荡器，而是外部晶体振荡后产生的高精度时钟了（接在 XTAL1/XTAL2 管脚上），也可以直接从 XTAL1 脚输入外部时钟，XTAL2 脚浮空。用户以后外部必须接晶体或时钟单片机才可以工作。

如果已被设置成用外部晶体或时钟工作的单片机，还要再设回使用内部 R/C 振荡器工作，则需给单片机外接晶体或时钟，再对 STC12C5201AD 系列单片机进行 ISP 下载用户程序时在选项中选择：



2.13 时钟分频及分频寄存器

时钟分频寄存器，可将时钟分成较低频率工作

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CLK_DIV	97h	Clock Divder	-	-	-	-	-	CLKS2	CLKS1	CLKS0	xxxx,x000

CLKS2	CLKS1	CLKS0	分频后CPU的实际工作时钟
0	0	0	系统时钟(外部时钟或内部R/C振荡时钟)
0	0	1	系统时钟/2
0	1	0	系统时钟/4
0	1	1	系统时钟/8
1	0	0	系统时钟/16
1	0	1	系统时钟/32
1	1	0	系统时钟/64
1	1	1	系统时钟/128

2.14 可编程时钟输出

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
WAKE_CLKO	8Fh	CLK_Output Powerdown_Wakeup Control Register	PCAWAKEUP	RXD_PIN_IE	T1_PIN_IE	T0_PIN_IE	LVD_WAKE	BRTCLKO	T1CLKO	T0CLKO	0000,0x00
BRT	9Ch	dedicated Baud-Rate Timer									0000,0000

sfr WAKE_CLKO = 0x8F; 新增加的特殊功能寄存器

/*

如何利用 CLKOUT0/P3.4 和 CLKOUT1/P3.5 管脚输出时钟

CLKOUT0/P3.4 和 CLKOUT1/P3.5 的时钟输出控制由 WAKE_CLKO 寄存器的 T0CLKO 位和 T0CLK1 位控制
CLKOUT0 的输出时钟频率由定时器 0 控制,CLKOUT1 的输出时钟频率由定时器 1 控制，相应的定时器需要工作在定时器的模式 2 方式(8 位自动重装载模式),不要允许相应的定时器中断,免得 CPU 反复进中断.

新增加的特殊功能寄存器：WAKE_CLKO （地址：0x8F)

b7 - PCAWAKEUP ： 允许 PCA 上升沿 / 下降沿中断 唤醒 powerdown。

b6 - RXD_PIN_IE: 允许 P3.0(RXD) 下降沿置 RI，也能使 RXD 唤醒 powerdown。

b5 - T1_PIN_IE ： 允许 T1/P3.5 脚下降沿置 T1 中断标志，也能使 T1 脚唤醒 powerdown。

b4 - T0_PIN_IE ： 允许 T0/P3.4 脚下降沿置 T0 中断标志，也能使 T0 脚唤醒 powerdown。

b3 - LVD_WAKE :1，允许在掉电模式下，LVD/P1.2 低压检测中断唤醒 CPU(STC12C5201AD 系列)。
允许在掉电模式下，LVD/P4.6 低压检测中断唤醒 CPU(STC12C5A60S2 系列)。

b2 - BRTCLKO :1，允许 P1.0 脚输出时钟，输出时钟频率 = 1/2 BRT 溢出率
BRT 工作在 1T 模式时的输出频率 CLKOUT2 =(Fosc / 2) / (256 - BRT)
BRT 工作在 12T 模式时的输出频率 CLKOUT2 =(Fosc / 2) / 12 / (256 - BRT)
0，不允许 BRT 在 P1.0 脚输出时钟

b1 - T1CLKO :1， 允许 T1 脚输出 T1(P3.5) 溢出脉冲，输出时钟频率 = 1/2 T1 溢出率
T1 工作在 1T 模式时的输出频率 CLKOUT1 =(Fosc / 2) / (256 - TH1)
T1工作在12T模式时的输出频率CLKOUT1 =(Fosc / 2) / 12 / (256 - TH1)
0，不允许 T1 脚输出 T1(P3.5) 溢出脉冲

b0 - T0CLKO :1， 允许 T0 脚输出 T0(P3.4) 溢出脉冲，输出时钟频率 = 1/2 T0 溢出率
T0 工作在 1T 模式时的输出频率 CLKOUT0 =(Fosc / 2) / (256 - TH0)
T0工作在12T模式时的输出频率CLKOUT0 =(Fosc / 2) / 12 / (256 - TH0)
0，不允许 T0 脚输出 T0(P3.4) 溢出脉冲

*/

如何利用 CLKOUT2/P1.0 管脚输出时钟(只针对 12C5A60S2/AD/PWM 系列)

CLKOUT2/P1.0 的时钟输出频率:
BRTx12 = 1,独立波特率发生器工作在 1T 模式
CLKOUT2 工作在 1T 模式时的输出频率 CLKOUT2 =(Fosc / 2) / (256 - BRT)
BRTx12 = 0,独立波特率发生器工作在 12T 模式
CLKOUT2 工作在 12T 模式时的输出频率 CLKOUT2 =(Fosc / 2) / 12 / (256 - BRT)

用户在程序中如何具体设置 CLKOUT2/P1.0 管脚输出时钟

1.对 BRT 寄存器独立波特率发生器定时器送 8 位重装载值，BRT = #reload_data
2.对 AUXR 寄存器中的 BRTR 位置 1,让独立波特率发生器定时器运行
3.对 WAKE_CLKO 寄存器中的 BRTCLKO 位置 1,让独立波特率发生器定时器的溢出在 P1.0 口输出时钟

```
/*本程序演示CLKOUT0/INT/T0/P3.4，CLKOUT1/INT/T1/P3.5，CLKOUT2/P1.0输出时钟演示程序*/
/*时钟频率 Fosc = 18.432MHz,T0, T1,独立波特率发生器均工作在12T模式*/
#include"reg51.h"
sfr WAKE_CLKO    = 0x8F;
sfr AUXR         = 0x8E;
sfr   BRT        = 0x9C;
main()
{
/*附加的 SFR WAKE_CLKO (地址：0x8F)
b7 - PCAWAKEUP : 允许 PCA上升沿 / 下降沿中断 唤醒 powerdown。
b6 - RXD_PIN_IE: 1，允许 RxD/P3.0(或RxD/P1.6) 下降沿置 RI，也能使 RxD脚唤醒 powerdown。
b5 - T1_PIN_IE : 1，允许 T1/P3.5 脚下降沿置 T1 中断标志，也能使 T1 脚唤醒 powerdown。
b4 - T0_PIN_IE : 1，允许 T0/P3.4 脚下降沿置 T0 中断标志，也能使 T0 脚唤醒 powerdown。
b3 - N/A
b2 - BRTCLKO :1，允许P1.0脚输出时钟，输出时钟频率 = 1/2 BRT 溢出率
                BRT 工作在1T模式时的输出频率CLKOUT2  =( Fosc / 2 ) / ( 256 - BRT )
                BRT 工作在12T模式时的输出频率CLKOUT2 =( Fosc / 2 ) / 12 / ( 256 - BRT )
             0，不允许BRT在P1.0脚输出时钟
b1 - T1CLKO  :1，允许 T1 脚输出 T1(P3.5) 溢出脉冲，输出时钟频率 = 1/2 T1 溢出率
                T1工作在1T模式时的输出频率CLKOUT1 =( Fosc / 2 ) / ( 256 - TH1 )
                T1工作在12T模式时的输出频率CLKOUT1 =( Fosc / 2 ) / 12 / ( 256 - TH1 )
             0，不允许 T1 脚输出 T1(P3.5) 溢出脉冲
b0 - T0CLKO  :1，允许 T0 脚输出 T0(P3.4) 溢出脉冲，输出时钟频率 = 1/2 T0 溢出率
                T0工作在1T模式时的输出频率CLKOUT0 =( Fosc / 2 ) / ( 256 - TH0 )
                T0工作在12T模式时的输出频率CLKOUT0 =( Fosc / 2 ) / 12 / ( 256 - TH0 )
             0，不允许 T0 脚输出 T0(P3.4) 溢出脉冲
*/

    TMOD = 0x22;               //T0，T1工作在模式2，8 位自动重装计数器
    AUXR = (AUXR | 0x80);      //T0工作在1T模式
    AUXR = (AUXR | 0x40);      // T1工作在1T模式
    AUXR = (AUXR | 0x04);      // 独立波特率发生器工作在1T模式

    BRT = (256-74);//对BRT独立波特率发生器定时器送8位重装载值，输出时钟频率124.540KHz
    TH0 = (256-74);//对T0做时钟输出的8位重装载数，18432000/2/74 = 124540.54约等于125K
    TH1 = (256-240); //对T1做时钟输出的8位重装载数,输出时钟频率18432000/2/240 = 38400

    WAKE_CLKO = ( WAKE_CLKO | 0x07);  允许T0,T1,独立波特率发生器输出时钟

    TR0 = 1；        //启动T0开始计数工作，对系统时钟进行分频输出
    TR1 = 1；        //启动T1开始计数工作，对系统时钟进行分频输出
    AUXR = (AUXR | 0x10);     //启动独立波特率发生器开始计数工作，对系统时钟进行分频输出
//至此时钟已经输出，用户可以通过示波器观看到输出的时钟频率
   while(1);
}
```

2.15 新增额外外部中断，及可将 CPU 从掉电模式唤醒的管脚

```
;/* --- STC International Limited ---------------------------------------------------- */
;/* --- 宏晶科技 姚永平 设计 2006/1/6 V1.0 ---------------------------------------- */
;/* --- 演示 STC12C5201AD 系列 MCU 从掉电模式唤醒 ------------------------------------ */
;/* --- Mobile: 13922805190 ---------------------------------------------------------- */
;/* --- Fax: 0755-82944243 ----------------------------------------------------------- */
;/* --- Tel: 0755-82948409 ----------------------------------------------------------- */
;/* --- Web: www.mcu-memory.com ------------------------------------------------------ */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;*************************************************************************
;Wake Up Idle and Wake Up Power Down
;*************************************************************************
;定义 STC12C5201AD 系列 MCU 特殊功能寄存器
#include "STC12C5201AD.H"
;-------------------------------------------------------------------------
;定义特殊功能寄存器
WAKE_CLKO    EQU 8FH
;附加的 SFR WAKE_CLKO (地址：0x8F)
;b7 - PCAWAKEUP :1, 允许 PCA 上升沿 / 下降沿中断  唤醒 powerdown。
;b6 - RXD_PIN_IE:1, 允许 P3.0(RXD) 下降沿置 RI, 也能使 RXD 唤醒 powerdown。
;b5 - T1_PIN_IE :1, 允许 T1/P3.5 脚下降沿置 T1 中断标志, 也能使 T1 脚唤醒 powerdown。
;b4 - T0_PIN_IE :1, 允许 T0/P3.4 脚下降沿置 T0 中断标志, 也能使 T0 脚唤醒 powerdown。
;b3 - LVD_WAKE  :1, 允许在掉电模式下, LVD/P1.2 低压检测中断唤醒 CPU(STC12C5201AD 系列)。
                    允许在掉电模式下, LVD/P4.6 低压检测中断唤醒 CPU(STC12C5A60S2 系列)。
                   0, 不许在掉电模式下, LVD/P1.2 低压检测中断唤醒 CPU(STC12C5201AD 系列)。
                    不允许在掉电模式下, LVD/P4.6 低压检测中断唤醒 CPU(STC12C5A60S2 系列)。
;b2 - BRTCLKO   :1, 允许 P1.0 脚输出时钟, 输出时钟频率 = 1/2 BRT 溢出率
                  BRT 工作在 1T 模式时的输出频率 CLKOUT2  =( Fosc / 2 ) / ( 256 - BRT )
                  BRT 工作在 12T 模式时的输出频率 CLKOUT2 =( Fosc / 2 ) / 12/( 256 - BRT )
                 0, 不允许 BRT 在 P1.0 脚输出时钟
;b1 - T1CLKO    :1, 允许 T1 脚输出 T1(P3.5) 溢出脉冲, Fck1 = 1/2 T1 溢出率
                0, 不允许 T1 脚输出 T1(P3.5) 溢出脉冲
;b0 - T0CLKO    :1, 允许 T0 脚输出 T0(P3.4) 溢出脉冲, Fck0 = 1/2 T1 溢出率
                0, 不允许 T0 脚输出 T0(P3.4) 溢出脉冲
;-------------------------------------------------------------------------
```

2.16 外部掉电检测专用比较器

---- 外部低压检测，增加了外部低压检测比较功能，可产生中断

STC12C5201AD 系列单片机在 P1.2 口增加了外部低压检测功能
STC12C5A60S2 系列单片机在 P4.6 口增加了外部低压检测功能
这样用户可以用查询方式或中断方式检查外部电压是否偏低。5伏单片机内部检测门槛电压是1.32V(+/-5%)，3 伏单片机内部检测门槛电压是 1.30V(+/-3%).
上电复位后外部低压检测标志位是 1，要由软件清零（注意该位不可位寻址），建议清零后，再读一次该位是否为零，如为零，才代表 P1.2 口的外部电压高于检测门槛电压。

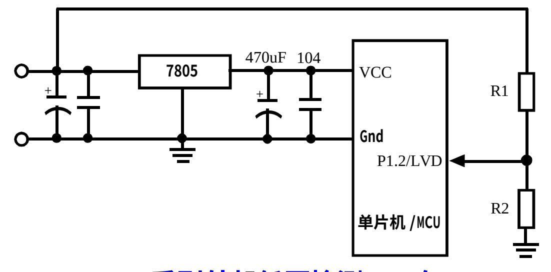
相应的中断控制允许位是： EA/ELVD，ELVD 是低压检测中断允许位
相应的中断优先级控制位是： PLVDH/PLVD,0/0,0/1,1/0,1/1,四级中断优先级
相应的中断请求标志位是： LVDF，要由软件清零

如果要求在掉电模式下外部低压检测中断继续工作,可将CPU从掉电模式唤醒,应将特殊功能寄存器WAKE_CLKO中的相应控制位 LVD_WAKE 置 1.如果不需要在掉电模式下外部低压检测中断唤醒,建议将 P1ASF 寄存器中的第二位 P12ASF 置 1,否则在掉电模式时,此低压检测电路会继续有几十 uA 的电流消耗.
当P1.2口作为外部低压检测时,如设置P12ASF位=1,在掉电模式时,该电路没有电流消耗,除非LVD_WAKE = 1,
当 P1 口中的相应位作为 A/D 使用时,要将 P1ASF 中的相应位置 1.

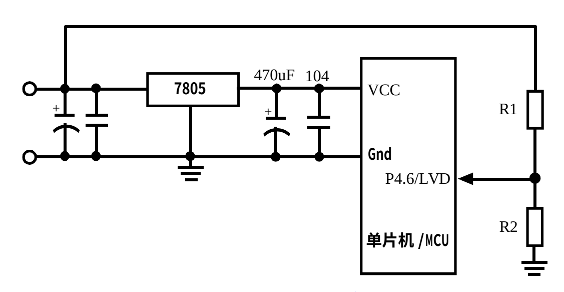
与外部低压检测 LVD 有关的特殊功能寄存器表

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
WAKE_CLKO	BFh	Clk_output Powerdown_Wakeup Control register	PCAWAKEUP	RXD_PIN_IE	T1_PIN_IE	T0_PIN_IE	LVD_WAKE	BRTCLKO	T1CLKO	T0CLKO	0000,0x00
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

利用增加的外部低压检测 LVD 功能作外部低压检测，典型应用线路图



STC12C5201AD 系列外部低压检测 LVD 在 P1.2 口



STC12C5A60S2 系列外部低压检测 LVD 在 P4.6 口

如交流电在 220V 时，稳压块 7805 前端的直流电是 11V,当交流电降到 160V 时，稳压块 7805 前端的直流电是 8.5V，图中的电阻 R1 和 R2 将 8.5V 的电压分压到低于低压检测门槛电压。此时 CPU 可以用查询方式查询，推荐使用中断，在中断服务程序里面，将 LVDF 位清零，再读 LVDF 位。如果为 0，则认为是电源抖动，如果为 1，则认为电源掉电，立即进行保存现场数据的工作。保存现场完成后，再将 LVDF 位清零，再读 LVDF 位的值。如果为 0，则认为电源系统恢复正常，此时 CPU 可恢复正常工作，如果为 1，继续将 LVDF 位清 0，再读 LVDF 的值，用此方法，等到电源恢复正常，或电源彻底掉电，CPU 进入复位状态。

```
;本程序用查询方式演示 P1.2 口外部低压检测
;IE:   EA,   ELVD, EADC, ES, ET1, EX1, ET0, EX0
;IP:   PPCA, PLVD, PADC, PS, PT1, PX1, PT1  PX0
;IPH:  PPCAH,PLVDH,PADCH,PSH,PT1H,PX1H,PT0H,PX0H
;PCON: SMOD, SMOD0,LVDF, POF,GF1, GF0, PD,  IDL
    P1ASF EQU  9DH
    ORG   0000H
    AJMP  MAIN
    ORG   0100H
MAIN:
    MOV SP, #0E0H    ;堆栈指针指向 0E0H 单元
    MOV     P1,#0F0H ;演示程序开始工作
    LCALL    Delay        ;延时
    MOV     P1,#0FH       ;演示程序开始工作
    LCALL    Delay        ;延时
    MOV      P1, #0FFH
MAIN1:
    MOV   P1ASF, #00000100B;  P1.2 口为模拟功能口
    MOV    A,PCON
    JBC    ACC.5,  POWER_ON_1
    CLR    P3.7     //ERROR_LED
    SETB  P3.5
    SETB  P3.4
    SETB  P3.3
ERROR:
    SJMP  ERROR
POWER_ON_1:
    SETB  P3.7
    CLR    P3.5     //POWER_ON_LED
    SETB  P3.4
    SETB  P3.3
    LCALL    Delay        ;延时
Continue_Read:
    MOV    A,#11011111B
    ANL    PCON, A
    NOP
    MOV    A,PCON
    JBC    ACC.5,  Low_Voltage
High_Voltage:
    SETB  P3.7
    SETB  P3.5
    CLR   P3.4    //High_Voltage_LED
    SETB  P3.3
    SJMP  Continue_Read
Low_Voltage:
    SETB  P3.7
    SETB  P3.5
    SETB  P3.4
    CLR    P3.3   //Low_Voltage_LED
    SJMP  Continue_Read
Delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #30H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET
;-----------------------------------------------
    END
```

2.17 STC12C5A60S2系列单片机内部扩展 1024字节RAM的使用

1).低128字节的内部RAM（地址：00H～7FH），可直接寻址或间接寻址，(data/idata)
2).高128字节的内部RAM（地址：80H～FFH），只能间接寻址（普通89C51没有），(idata)
3).特殊功能寄存器SFR（地址：80H～FFH），只能直接寻址，(data)

特殊功能寄存器SFR和高128字节的内部RAM是通过寻址方式来区分的，传统的8051系列单片机只有128-256字节RAM供用户使用，在此情况下STC公司响应广大用户的呼声，在一些单片机内部增加了扩展RAM。STC12C5A60S2/AD/PWM系列单片机内部扩展了1024个字节RAM，共1280字节RAM。访问内部扩展RAM时，不影响P0口/P2口/P3.6/P3.7/ALE。

STC12C5A60S2/AD/PWM系列单片机8051 单片机 扩展RAM管理及禁止ALE输出 特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000

Symbol 符号　Function 功能
EXTRAM　Internal/External RAM access 内部/外部 RAM 存取
0: 内部扩展的EXT_RAM可以存取.
STC12C5A60S2/AD/PWM系列单片机
在00H到3FFH单元(1024字节),使用MOVX @DPTR指令访问,超过400H的地址空间总是访问外部数据存储器（含400H单元），MOVX @Ri只能访问00H到FFH单元

1: External data memory access.
外部数据存储器存取,禁止访问内部扩展RAM，此时MOVX @DPTR / MOVX @Ri的使用同普通8052单片机

应用示例供参考（汇编）:

访问内部扩展的 EXTRAM

```
;新增特殊功能寄存器声明(汇编方式)
AUXR DATA 8EH;          或者用    AUXR   EQU   8EH    定义
MOV     AUXR,    #00000000B;   EXTRAM 位清为”0”， 其实上电复位时此位就为”0”.
    ;MOVX  A,  @DPTR / MOVX   @DPTR,  A 指令可访问内部扩展的 EXTRAM
        ;RD+ 系列为(00H - 3FFH,共 1024 字节)
        ;RC 系列为(00H - FFH,共 256 字节)
    ;MOVX  A,  @Ri / MOVX  A, @Ri  指令可直接访问内部扩展的 EXTRAM
        ;使用此指令 RD+ 系列 只能访问内部扩展的 EXTRAM(00H - FFH,共 256 字节)
;写芯片内部扩展的 EXTRAM
    MOV     DPTR,    #address
    MOV     A,   #value
    MOVX @DPTR,  A
;读芯片内部扩展的 EXTRAM
    MOV     DPTR,    #address
    MOVX A,  @DPTR
```

RD+ 系列

; 如果 #address < 400H，则在 EXTRAM 位为”0”时，访问物理上在内部，逻辑上在外部的此 EXTRAM

; 如果 #address >= 400H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（400H--FFFFH）

禁止访问内部扩展的 EXTRAM ,以防冲突

MOV AUXR, #00000010B; EXTRAM 控制位设置为”1”，禁止访问 EXTRAM,以防冲突

有些用户系统因为外部扩展了 I/O 或者用片选去选多个 RAM 区,有时与此内部扩展的 EXTRAM 逻辑地址上有冲突，将此位设置为”1”，禁止访问此内部扩展的 EXTRAM 就可以了.

大实话： 其实不用设置 AUXR 寄存器即可直接用 MOVX @DPTR 指令访问此内部扩展的 EXTRAM,超过此 RAM 空间,将访问片外单元.如果系统外扩了 SRAM,而实际使用的空间小于 1024 字节,则可直接将此 SRAM 省去,比如省去 STC62WV256,IS62C256,UT6264 等.

应用示例供参考（C 语言）:

```
/*  访问内部扩展的 EXTRAM */
/*  STC12C5A60S2/AD/PWM 系列单片机为(00H - 3FFH，共 1024 字节扩展的 EXTRAM) */
    /*  新增特殊功能寄存器声明(C 语言方式)     */
    sfr AUXR=   0x8e /*  如果不需设置 AUXR 就不用声明 AUXR       */
    AUXR=   0x00;   /*  0000,0000  EXTRAM 位清 0，其实上电复位时此位就为 0 */
    unsigned char  xdata  sum,   loop_counter, test_array[128];
    /*  将变量声明成 xdata 即可直接访问此内部扩展的 EXTRAM */
```

```
/*  写芯片内部扩展的 EXTRAM  */
        sum =    0;
        loop_counter =    128;
        test_array[0]    =    5;

/*  读芯片内部扩展的 EXTRAM */
        sum =    test_array[0];
    /*  RD+ 系列:
                如果 #address < 400H，则在 EXTRAM 位为"0"时，访问物理上在内部，逻辑
                 上在外部的此 EXTRAM
                如果 #address>=400H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（400H-FFFFH）
    */
```

禁止访问内部扩展的 EXTRAM，以防冲突

```
AUXR=    0x02;    /*   0000,0010， EXTRAM 位设为"1"，禁止访问 EXTRAM,以防冲突  */
```

有些用户系统因为外部扩展了 I/O 或者用片选去选多个 RAM 区,有时与此内部扩展的 EXTRAM 逻辑上有冲突，将此位设置为"1"，禁止访问此内部扩展的 EXTRAM 就可以了.

STC12C5A60S2 系列单片机内部扩展 RAM 演示程序

```
;/* --- STC International Limited ---------------------------------------- */
;/* --- 宏晶科技 姚永平  设计 2006/1/6   V1.0 ---------------------------- */
;/* --- 演示 STC12C5A60S2/AD/PWM 系列单片机 MCU 内部扩展 RAM 演示程序 -------- */
;/* --- Mobile: 13922805190 --------------------------------------------- */
;/* --- Fax: 0755-82944243 ---------------------------------------------- */
;/* --- Tel: 0755-82948409 ---------------------------------------------- */
;/* --- Web: www.mcu-memory.com ----------------------------------------- */
;/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 --- */
;/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --- */

#include <reg52.h>
#include <intrins.h>        /* use _nop_() function */

sfr AUXR = 0x8e;

sbit ERROR_LED = P1^5;
sbit OK_LED = P1^7;

void main()
{
    unsigned int array_point = 0;

    /* 测试数组 Test_array_one[512],Test_array_two[512]*/
    unsigned char xdata Test_array_one[512]        =
    {
        0x00,     0x01,     0x02,     0x03,     0x04,     0x05,     0x06,     0x07,
        0x08,     0x09,     0x0a,     0x0b,     0x0c,     0x0d,     0x0e,     0x0f,
        0x10,     0x11,     0x12,     0x13,     0x14,     0x15,     0x16,     0x17,
        0x18,     0x19,     0x1a,     0x1b,     0x1c,     0x1d,     0x1e,     0x1f,
        0x20,     0x21,     0x22,     0x23,     0x24,     0x25,     0x26,     0x27,
```

```
0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f,
0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f,
0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f,
0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77,
0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f,
0x80, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,
0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x8d, 0x8e, 0x8f,
0x90, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, 0x97,
0x98, 0x99, 0x9a, 0x9b, 0x9c, 0x9d, 0x9e, 0x9f,
0xa0, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7,
0xa8, 0xa9, 0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf,
0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7,
0xb8, 0xb9, 0xba, 0xbb, 0xbc, 0xbd, 0xbe, 0xbf,
0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,
0xd0, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, 0xd7,
0xd8, 0xd9, 0xda, 0xdb, 0xdc, 0xdd, 0xde, 0xdf,
0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7,
0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef,
0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
0xf8, 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff,
0xff, 0xfe, 0xfd, 0xfc, 0xfb, 0xfa, 0xf9, 0xf8,
0xf7, 0xf6, 0xf5, 0xf4, 0xf3, 0xf2, 0xf1, 0xf0,
0xef, 0xee, 0xed, 0xec, 0xeb, 0xea, 0xe9, 0xe8,
0xe7, 0xe6, 0xe5, 0xe4, 0xe3, 0xe2, 0xe1, 0xe0,
0xdf, 0xde, 0xdd, 0xdc, 0xdb, 0xda, 0xd9, 0xd8,
0xd7, 0xd6, 0xd5, 0xd4, 0xd3, 0xd2, 0xd1, 0xd0,
0xcf, 0xce, 0xcd, 0xcc, 0xcb, 0xca, 0xc9, 0xc8,
0xc7, 0xc6, 0xc5, 0xc4, 0xc3, 0xc2, 0xc1, 0xc0,
0xbf, 0xbe, 0xbd, 0xbc, 0xbb, 0xba, 0xb9, 0xb8,
0xb7, 0xb6, 0xb5, 0xb4, 0xb3, 0xb2, 0xb1, 0xb0,
0xaf, 0xae, 0xad, 0xac, 0xab, 0xaa, 0xa9, 0xa8,
0xa7, 0xa6, 0xa5, 0xa4, 0xa3, 0xa2, 0xa1, 0xa0,
0x9f, 0x9e, 0x9d, 0x9c, 0x9b, 0x9a, 0x99, 0x98,
0x97, 0x96, 0x95, 0x94, 0x93, 0x92, 0x91, 0x90,
0x8f, 0x8e, 0x8d, 0x8c, 0x8b, 0x8a, 0x89, 0x88,
0x87, 0x86, 0x85, 0x84, 0x83, 0x82, 0x81, 0x80,
0x7f, 0x7e, 0x7d, 0x7c, 0x7b, 0x7a, 0x79, 0x78,
0x77, 0x76, 0x75, 0x74, 0x73, 0x72, 0x71, 0x70,
0x6f, 0x6e, 0x6d, 0x6c, 0x6b, 0x6a, 0x69, 0x68,
0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61, 0x60,
0x5f, 0x5e, 0x5d, 0x5c, 0x5b, 0x5a, 0x59, 0x58,
```

```
        0x57,     0x56,     0x55,     0x54,     0x53,     0x52,     0x51,     0x50,
        0x4f,     0x4e,     0x4d,     0x4c,     0x4b,     0x4a,     0x49,     0x48,
        0x47,     0x46,     0x45,     0x44,     0x43,     0x42,     0x41,     0x40,
        0x3f,     0x3e,     0x3d,     0x3c,     0x3b,     0x3a,     0x39,     0x38,
        0x37,     0x36,     0x35,     0x34,     0x33,     0x32,     0x31,     0x30,
        0x2f,     0x2e,     0x2d,     0x2c,     0x2b,     0x2a,     0x29,     0x28,
        0x27,     0x26,     0x25,     0x24,     0x23,     0x22,     0x21,     0x20,
        0x1f,     0x1e,     0x1d,     0x1c,     0x1b,     0x1a,     0x19,     0x18,
        0x17,     0x16,     0x15,     0x14,     0x13,     0x12,     0x11,     0x10,
        0x0f,     0x0e,     0x0d,     0x0c,     0x0b,     0x0a,     0x09,     0x08,
        0x07,     0x06,     0x05,     0x04,     0x03,     0x02,     0x01,     0x00
    };

    unsigned char xdata Test_array_two[512]        =
    {
        0x00,     0x01,     0x02,     0x03,     0x04,     0x05,     0x06,     0x07,
        0x08,     0x09,     0x0a,     0x0b,     0x0c,     0x0d,     0x0e,     0x0f,
        0x10,     0x11,     0x12,     0x13,     0x14,     0x15,     0x16,     0x17,
        0x18,     0x19,     0x1a,     0x1b,     0x1c,     0x1d,     0x1e,     0x1f,
        0x20,     0x21,     0x22,     0x23,     0x24,     0x25,     0x26,     0x27,
        0x28,     0x29,     0x2a,     0x2b,     0x2c,     0x2d,     0x2e,     0x2f,
        0x30,     0x31,     0x32,     0x33,     0x34,     0x35,     0x36,     0x37,
        0x38,     0x39,     0x3a,     0x3b,     0x3c,     0x3d,     0x3e,     0x3f,
        0x40,     0x41,     0x42,     0x43,     0x44,     0x45,     0x46,     0x47,
        0x48,     0x49,     0x4a,     0x4b,     0x4c,     0x4d,     0x4e,     0x4f,
        0x50,     0x51,     0x52,     0x53,     0x54,     0x55,     0x56,     0x57,
        0x58,     0x59,     0x5a,     0x5b,     0x5c,     0x5d,     0x5e,     0x5f,
        0x60,     0x61,     0x62,     0x63,     0x64,     0x65,     0x66,     0x67,
        0x68,     0x69,     0x6a,     0x6b,     0x6c,     0x6d,     0x6e,     0x6f,
        0x70,     0x71,     0x72,     0x73,     0x74,     0x75,     0x76,     0x77,
        0x78,     0x79,     0x7a,     0x7b,     0x7c,     0x7d,     0x7e,     0x7f,
        0x80,     0x81,     0x82,     0x83,     0x84,     0x85,     0x86,     0x87,
        0x88,     0x89,     0x8a,     0x8b,     0x8c,     0x8d,     0x8e,     0x8f,
        0x90,     0x91,     0x92,     0x93,     0x94,     0x95,     0x96,     0x97,
        0x98,     0x99,     0x9a,     0x9b,     0x9c,     0x9d,     0x9e,     0x9f,
        0xa0,     0xa1,     0xa2,     0xa3,     0xa4,     0xa5,     0xa6,     0xa7,
        0xa8,     0xa9,     0xaa,     0xab,     0xac,     0xad,     0xae,     0xaf,
        0xb0,     0xb1,     0xb2,     0xb3,     0xb4,     0xb5,     0xb6,     0xb7,
        0xb8,     0xb9,     0xba,     0xbb,     0xbc,     0xbd,     0xbe,     0xbf,
        0xc0,     0xc1,     0xc2,     0xc3,     0xc4,     0xc5,     0xc6,     0xc7,
        0xc8,     0xc9,     0xca,     0xcb,     0xcc,     0xcd,     0xce,     0xcf,
        0xd0,     0xd1,     0xd2,     0xd3,     0xd4,     0xd5,     0xd6,     0xd7,
        0xd8,     0xd9,     0xda,     0xdb,     0xdc,     0xdd,     0xde,     0xdf,
        0xe0,     0xe1,     0xe2,     0xe3,     0xe4,     0xe5,     0xe6,     0xe7,
        0xe8,     0xe9,     0xea,     0xeb,     0xec,     0xed,     0xee,     0xef,
        0xf0,     0xf1,     0xf2,     0xf3,     0xf4,     0xf5,     0xf6,     0xf7,
        0xf8,     0xf9,     0xfa,     0xfb,     0xfc,     0xfd,     0xfe,     0xff,
        0xff,     0xfe,     0xfd,     0xfc,     0xfb,     0xfa,     0xf9,     0xf8,
```

```
        0xf7,     0xf6,     0xf5,     0xf4,     0xf3,     0xf2,     0xf1,     0xf0,
        0xef,     0xee,     0xed,     0xec,     0xeb,     0xea,     0xe9,     0xe8,
        0xe7,     0xe6,     0xe5,     0xe4,     0xe3,     0xe2,     0xe1,     0xe0,
        0xdf,     0xde,     0xdd,     0xdc,     0xdb,     0xda,     0xd9,     0xd8,
        0xd7,     0xd6,     0xd5,     0xd4,     0xd3,     0xd2,     0xd1,     0xd0,
        0xcf,     0xce,     0xcd,     0xcc,     0xcb,     0xca,     0xc9,     0xc8,
        0xc7,     0xc6,     0xc5,     0xc4,     0xc3,     0xc2,     0xc1,     0xc0,
        0xbf,     0xbe,     0xbd,     0xbc,     0xbb,     0xba,     0xb9,     0xb8,
        0xb7,     0xb6,     0xb5,     0xb4,     0xb3,     0xb2,     0xb1,     0xb0,
        0xaf,     0xae,     0xad,     0xac,     0xab,     0xaa,     0xa9,     0xa8,
        0xa7,     0xa6,     0xa5,     0xa4,     0xa3,     0xa2,     0xa1,     0xa0,
        0x9f,     0x9e,     0x9d,     0x9c,     0x9b,     0x9a,     0x99,     0x98,
        0x97,     0x96,     0x95,     0x94,     0x93,     0x92,     0x91,     0x90,
        0x8f,     0x8e,     0x8d,     0x8c,     0x8b,     0x8a,     0x89,     0x88,
        0x87,     0x86,     0x85,     0x84,     0x83,     0x82,     0x81,     0x80,
        0x7f,     0x7e,     0x7d,     0x7c,     0x7b,     0x7a,     0x79,     0x78,
        0x77,     0x76,     0x75,     0x74,     0x73,     0x72,     0x71,     0x70,
        0x6f,     0x6e,     0x6d,     0x6c,     0x6b,     0x6a,     0x69,     0x68,
        0x67,     0x66,     0x65,     0x64,     0x63,     0x62,     0x61,     0x60,
        0x5f,     0x5e,     0x5d,     0x5c,     0x5b,     0x5a,     0x59,     0x58,
        0x57,     0x56,     0x55,     0x54,     0x53,     0x52,     0x51,     0x50,
        0x4f,     0x4e,     0x4d,     0x4c,     0x4b,     0x4a,     0x49,     0x48,
        0x47,     0x46,     0x45,     0x44,     0x43,     0x42,     0x41,     0x40,
        0x3f,     0x3e,     0x3d,     0x3c,     0x3b,     0x3a,     0x39,     0x38,
        0x37,     0x36,     0x35,     0x34,     0x33,     0x32,     0x31,     0x30,
        0x2f,     0x2e,     0x2d,     0x2c,     0x2b,     0x2a,     0x29,     0x28,
        0x27,     0x26,     0x25,     0x24,     0x23,     0x22,     0x21,     0x20,
        0x1f,     0x1e,     0x1d,     0x1c,     0x1b,     0x1a,     0x19,     0x18,
        0x17,     0x16,     0x15,     0x14,     0x13,     0x12,     0x11,     0x10,
        0x0f,     0x0e,     0x0d,     0x0c,     0x0b,     0x0a,     0x09,     0x08,
        0x07,     0x06,     0x05,     0x04,     0x03,     0x02,     0x01,     0x00
    };
    ERROR_LED = 1;
    OK_LED = 1;
    for(array_point=0; array_point<512; array_point++)
    {
        if(Test_array_one[array_point]!=Test_array_two [array_point]){
            ERROR_LED = 0;
            OK_LED = 1;
            break;
        }
        else{
            OK_LED = 0;
            ERROR_LED = 1;
         }
    }
    while(1);
}
```

2.18 双数据指针 DPTR0,DPTR1 的使用

12C5A60PWM/AD/S2 系列 8051 单片机 双数据指针 特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1	A2h	Auxiliary Register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0

Symbol 符号 Function 功能

DPS DPTR registers select bit. DPTR 寄存器选择位
0: DPTR0 is selected DPTR0 被选择
1: DPTR1 is selected DPTR1 被选择

此系列单片机有两个 16-bit 数据指针,DPTR0,DPTR1.当 DPS 选择位为 0 时,选择 DPTR0, 当 DPS 选择位为 1 时,选择 DPTR1.

AUXR1 特殊功能寄存器,位于 A2H 单元,其中的位不可用布尔指令快速访问.但由于 DPS 位位于 bit0,故对 AUXR1 寄存器用 INC 指令,DPS 位便会反转,由 0 变成 1 或由 1 变成 0,即可实现双数据指针的快速切换.

应用示例供参考:

```
;新增特殊功能寄存器定义
AUXR1      DATA0A2H
MOV        AUXR1,   #0        ;此时 DPS 为 0,DPTR0 有效

MOV        DPTR,    #1FFH     ;置 DPTR0 为 1FFH
MOV        A,       #55H
MOVX@DPTR,    A          ;将 1FFH 单元置为 55H

MOV        DPTR,    #2FFH     ; 置 DPTR0 为 2FFH
MOV        A,       #0AAH
MOVX@DPTR,    A          ;将 2FFH 单元置为 0AAH

INC        AUXR1              ; 此时 DPS 为 1,DPTR1 有效
MOV        DPTR,    #1FFH     ; 置 DPTR1 为 1FFH
MOVXA,         @DPTR     ; 读 DPTR1 数据指针指向的 1FFH 单元的内容,累加器 A 变为 55H.

INC        AUXR1              ; 此时 DPS 为 0,DPTR0 有效
MOVXA,         @DPTR     ; 读 DPTR0 数据指针指向的 2FFH 单元的内容,累加器 A 变为 0AAH.

INC        AUXR1              ; 此时 DPS 为 1,DPTR1 有效
MOVXA,         @DPTR     ; 读 DPTR1 数据指针指向的 1FFH 单元的内容,累加器 A 变为 55H.

INC        AUXR1              ; 此时 DPS 为 0,DPTR0 有效
MOVXA,         @DPTR     ; 读 DPTR0 数据指针指向的 2FFH 单元的内容,累加器 A 变为 0AAH.
```

2.19 STC12C5A60AD系列单片机片外64K数据总线速度控制

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
BUS_SPEED	A1h	Bus-Speed Control			ALES1	ALES0		RWS2	RWS1	RWS0	xx10,x011

ALES1	ALES0	
0	0	P0地址建立时间和保持时间到ALE信号的下降沿是1个时钟
0	1	P0地址建立时间和保持时间到ALE信号的下降沿是2个时钟
1	0	P0地址建立时间和保持时间到ALE信号的下降沿是3个时钟(复位之后默认设置)
1	1	P0地址建立时间和保持时间到ALE信号的下降沿是4个时钟

RWS2	RWS1	RWS0	
0	0	0	MOVX 读/写 脉冲是1个时钟
0	0	1	MOVX 读/写 脉冲是2个时钟
0	1	0	MOVX 读/写 脉冲是3个时钟
0	1	1	MOVX 读/写 脉冲是4个时钟(复位之后默认设置)
1	0	0	MOVX 读/写 脉冲是5个时钟
1	0	1	MOVX 读/写 脉冲是6个时钟
1	1	0	MOVX 读/写 脉冲是7个时钟
1	1	1	MOVX 读/写 脉冲是8个时钟

当MOVX 指令访问物理上在内部，逻辑上在外部的片内扩展的1024字节EXT_RAM时，以上设置均被忽略，以上设置只是在访问真正的片外扩展器件时有效。

助记符	功能说明	字节数	1时钟/机器周期单片机所需时钟	效率提升
MOVX A,@Ri	逻辑上在外部的片内扩展RAM，(8位地址) 送入累加器	1	4	6倍
MOVX A,@DPTR	逻辑上在外部的片内扩展RAM，(16位地址) 送入累加器	1	3	8倍
MOVX @Ri,A	累加器送逻辑上在外部的片内扩展RAM (8位地址)	1	3	8倍
MOVX @DPTR ,A	累加器送逻辑上在外部的片内扩展RAM (16位地址)	1	3	8倍
MOVX A,@Ri	物理上在外部的片外扩展RAM，(8位地址) 送入累加器	1	7 + ?	*Note1
MOVX A,@DPTR	物理上在外部的片外扩展RAM，(16位地址) 送入累加器	1	7 + ?	*Note1
MOVX @Ri,A	累加器送物理上在外部的片外扩展RAM，(8位地址)	1	7 + ?	*Note1
MOVX @DPTR ,A	累加器送物理上在外部的片外扩展RAM，(16位地址)	1	7 + ?	*Note1

Note1:访问物理上在片外的扩展RAM所需时钟：7 + 2 x ALE_Bus_Speed + RW_Bus_Speed

其中ALE_Bus_Speed由BUS_SPEED控制寄存器中的ALES1/ALES0决定

其中RW_Bus_Speed由BUS_SPEED控制寄存器中的RWS2/RWS1/RWS0决定

2.20 STC12C5A60S2系列单片机P4/P5口的使用

STC12C5A60S2/AD/PWM系列单片机与P4/P5口有关的特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P4	C0h	8 - bit Port 4	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0	1111,1111
P4M1	B3h	P4 Configuration 1									0000,0000
P4M0	B4h	P4 Configuration 0									0000,0000
P4SW	BBh	Port - 4 switch	-	LVD_P4.6	ALE_P4.5	NA_P4.4	-	-	-	-	x000,xxxx
AUXR1	A2h	Auxiliary register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0
P5	C8h	8 - bit Port 5	-	-	-	-	P5.3	P5.2	P5.1	P5.0	xxxx,1111
P5M1	C9h	P5 Configuration 1									0000,0000
P5M0	CAh	P5 Configuration 0									0000,0000

对STC12C5A60S2/AD/PWM系列单片机的P4/P5口的访问，如同访问常规的P1/P2/P3口，并且均可位寻址，P4的地址C0H，P5口的地址在C8H。

P4端口的地址在C0h，P4口中的每一位均可位寻址，位地址如下：								
位	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
位地址	C7h	C6h	C5h	C4h	C3h	C2h	C1h	C0h

P5端口的地址在C8h，P5口中的每一位均可位寻址，位地址如下：								
位	-	-	-	-	P5.3	P5.2	P5.1	P5.0
位地址					CBh	CAh	C9h	C8h

由P4SW寄存器设置(NA/P4.4，ALE/P4.5，EX_LVD/P4.6)三个端口的第二功能

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P4SW	BBh	Port - 4 switch		LVD_P4.6	ALE_P4.5	NA_P4.4					x000,xxxx

NA/P4.4: 0,复位后P4SW.4 = 0,NA/P4.4脚是弱上拉，无任何功能
1,通过设置P4SW.4 = 1,将NA/P4.4脚设置成I/O口(P4.4)
ALE/P4.5: 0,复位后P4SW.5 = 0,ALE/P4.5脚是ALE信号,只有在用MOVX指令访问片外扩展器件时才有信号输出
1，通过设置P4SW.5 = 1,将ALE/P4.5脚设置成I/O口(P4.5)
EX_LVD/P4.6: 0，复位后P4SW.6 = 0,EX_LVD/P4.6是外部低压检测脚，可使用查询方式或设置成中断来检测
1,通过设置P4SW.6 = 1将EX_LVD/P4.6脚设置成I/O口(P4.6)

在ISP烧录程序时设置RST/P4.7的第二功能

RST/P4.7在ISP烧录程序时选择是复位脚还是P4.7口，如设置成P4.7口，必须使用外部时钟。

由AUXR1寄存器设置(PCA/PWM/SPI/UART2)是在P1口还是在P4口

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1	A2h	Auxiliary register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0

PCA_P4: 0,复位后AUXR1.6 = 0,PCA/PWM在P1口
1,通过设置AUXR1.6 = 1,将PCA/PWM从P1口切换到P4口
SPI_P4: 0,复位后AUXR1.5 = 0,SPI在P1口
1,通过设置AUXR1.5 = 1，将SPI从P1口切换到P4口
S2_P4: 0,复位后AUXR1.4 = 0,UART2在P1口(仅针对双串口单片机有效)
1,通过设置AUXR1.4 = 1，将UART2从P1口切换到P4口(仅针对双串口单片机有效)
具体的PCA/SPI/S2管脚是如何从P1口影射到P4口的，见单片机管脚图)

2.21 将STC12C5A60S2系列SPI/PCA/PWM/UART2从P1口设置到P4口

STC12C5A60PWM/12C5A60AD/12C5A60S2系列单片机的AUXR1寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1	A2h	Auxiliary Register 1	-	PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	x000,00x0

PCA_P4: 0，缺省PCA在P1口
1，PCA/PWM从P1口切换到P4口
ECI从P1.2切换到P4.1口
PCA0/PWM0从P1.3切换到P4.2口
PCA1/PWM1从P1.4切换到P4.3口

SPI_P4: 0，缺省SPI在P1口
1，SPI从P1口切换到P4口
SPICLK从P1.7切换到P4.3口
MISO从P1.6切换到P4.2口
MOSI从P1.5切换到P4.1口
SS从P1.4切换到P4.0口

S2_P4: 0，缺省UART2在P1口
1，UART2从P1口切换到P4口
TxD2从P1.3切换到P4.3口
RxD2从P1.2切换到P4.2口

GF2: 通用标志位

ADRJ: 0，10位A/D转换结果的高8位放在ADC_RES寄存器，
低2位放在ADC_RESL寄存器
1，10位A/D转换结果的最高2位放在ADC_RES寄存器的低2位，
低8位放在ADC_RESL寄存器

DPS: 0，使用缺省数据指针DPTR0
1，使用另一个数据指针DPTR1

2.22 串行口1使用独立波特率发生器作为波特率发生器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
BRT	9Ch	dedicated Baud-Rate timer 独立波特率发生器定时器,装入重装载数									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000
以下是使用定时器1作为串口1的波特率发生器时需要用到的寄存器，现在可以不用了											
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000

当设置AUXR寄存器中的S1BRS位(串口1波特率选择位)为1时，串口1选择独立波特率发生器作为波特率发生器,此时定时器1可以释放出来作为定时器/计数器/时钟输出使用.

SM0	SM1	方式	功能说明	串口1波特率
0	0	0	同步移位串行方式	Fosc / 12, UART_M0x6 = 1时，波特率是Fosc / 2
0	1	1	8位UART，波特率可变	(2^{SMOD} / 32) x BRT独立波特率发生器的溢出率
1	0	2	9位UART	(2^{SMOD} / 64) x Fosc系统工作时钟频率
1	1	3	9位UART，波特率可变	(2^{SMOD} / 32) x BRT独立波特率发生器的溢出率
BRT独立波特率发生器的溢出率 = Fosc/12/(256 - BRT),当BRTx12 = 0时 BRT独立波特率发生器的溢出率 = Fosc / (256 - BRT),当BRTx12 = 1时				

串口 1 模式 0：

串行数据通过 RxD/P3.0 接收,TxD/P3.1 输出同步移位时钟，发送接收的是八位数据,低位在先,波特率固定在 Fosc / 12，忽略波特率发生器

串口 1 波特率在模式 0 = Fosc 系统工作时钟频率 / 12

串口 1 模式 1：

10 位数据通过 TxD/P3.1 发送，通过 RxD/P3.0 接收。一帧数据包含一个起始位(0)，8 个数据位(低位在先)，和一个停止位(1)。接收时，停止位进入特殊功能寄存器 SCON 的 RB8 位。波特率由独立波特率发生器 BRT 的溢出率决定。

串口 1 波特率在模式 1 = (2^{SMOD} / 32) x BRT 独立波特率发生器的溢出率

当 SMOD = 0 时，串口 2 波特率 = BRT 独立波特率发生器的溢出率 / 32，

当 SMOD = 1 时，串口 2 波特率 = BRT 独立波特率发生器的溢出率 / 16，

BRT 独立波特率发生器的溢出率 = Fosc/12/(256 - BRT),当 BRTx12 = 0 时，

BRT 独立波特率发生器的溢出率 = Fosc / (256 - BRT),当 BRTx12 = 1 时

串口 1 模式 2：

11 位数据通过 TxD/P3.1 发送，通过 RxD/P3.0 接收。一帧数据包含一个起始位(0)，8 个数据位(低位在先)，一个可编程的第 9 位，和一个停止位(1)。发送时，第 9 位数据位来自特殊功能寄存器 SCON 的 TB8 位.接收时，第 9 位进入特殊功能寄存器 SCON 的 RB8 位。波特率可编程为系统时钟频率：Fosc / 32 或者 Fosc / 64，串口 2 工作在模式 2 和串口 1 工作在模式 2 是相同的。

串口 1 波特率在模式 2 = (2^{SMOD} / 64) x Fosc 系统工作时钟频率

当 SMOD = 0 时，串口 2 波特率 = Fosc 系统工作时钟频率 / 64

当 SMOD = 1 时，串口 2 波特率 = Fosc 系统工作时钟频率 / 32

串口 1 模式 3：

波特率是可变的，其它和模式 2 相同

11 位数据通过 TxD/P3.1 发送，通过 RxD/P3.0 接收。一帧数据包含一个起始位(0)，8 个数据位(低位在先)，一个可编程的第 9 位，和一个停止位(1)。发送时，第 9 位数据位来自特殊功能寄存器 SCON 的 TB8 位.接收时，第 9 位进入特殊功能寄存器 SCON 的 RB8 位。

串口 1 波特率在模式 3 = (2^{SMOD} / 32) x BRT 独立波特率发生器的溢出率

当 SMOD = 0 时，串口 1 波特率 = BRT 独立波特率发生器的溢出率 / 32，

当 SMOD = 1 时，串口 1 波特率 = BRT 独立波特率发生器的溢出率 / 16，

BRT 独立波特率发生器的溢出率 = Fosc/12/(256 - BRT),当 BRTx12 = 0 时，

BRT 独立波特率发生器的溢出率 = Fosc / (256 - BRT),当 BRTx12 = 1 时

用户在程序中如何具体使用串口 1 和独立波特率发生器 BRT

1.设置串口 1 的工作模式，SCON 寄存器中的 SM0 和 SM1 两位决定了串口 1 的 4 种工作模式。

2.设置串口 1 的波特率,使用独立波特率发生器寄存器和相应的位：

BRT 独立波特率发生器寄存器,BRTx12 位，SMOD 位

3.启动独立波特率发生器，让 BRTR 位为 1，BRT 独立波特率发生器寄存器就立即开始计数。

4.设置串口 1 的中断优先级，及打开中断相应的控制位是:

PS,PSH,ES,EA

5.如要串口 1 接收，将 REN 置 1 即可

如要串口 1 发送，将数据送入 SBUF 即可，

接收完成标志 RI,发送完成标志 TI,要由软件清 0。

```
;当串口工作在模式1和模式3时,计算相应的波特率需要设置的重装载数,结果送入BRT寄存器
;计算自动重装数 RELOAD (SMOD = 0, SMOD 是 PCON 特殊功能寄存器的最高位):
;  1. 计算 RELOAD (以下是 SMOD = 0 时的计算公式)
;
;     a) 12T 模式的计算公式: RELOAD = 256 - INT(Fosc/Baud0/32/12 + 0.5)
;     b)  1T 模式的计算公式: RELOAD = 256 - INT(Fosc/Baud0/32 + 0.5)
计算出的RELOAD数直接送BRT寄存器
;
;     式中: INT() 表示取整运算即舍去小数, 在式中加 0.5 可以达到四舍五入的目的
;           Fosc = 晶振频率
;           Baud0 = 标准波特率
;
;  2. 计算用 RELOAD 产生的波特率:
;       a)   Baud = Fosc/(256 - RELOAD)/32/12        12T 模式
;       b)   Baud = Fosc/(256 - RELOAD)/32            1T 模式
;
;  3. 计算误差
;          error = (Baud - Baud0)/Baud0 * 100%
;  4. 如果误差绝对值 > 3% 要更换波特率或者更换晶体频率, 重复步骤 1-4
;
;
;例: Fosc = 22.1184MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 22118400/57600/32/12 + 0.5)
;           = 256 - INT( 1.5 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 22118400/(256-255)/32/12
;          = 57600
; 3. 误差等于零
```

```
;例: Fosc = 18.432MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/57600/32/12 + 0.5)
;           = 256 - INT( 0.833 + 0.5 )
;           = 256 - INT( 1.333 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 18432000/(256-255)/32/12
;         = 48000
; 3. error = (48000 - 57600)/57600 * 100%
;          = -16.66%
; 4. 误差很大，要更换波特率或者更换晶体频率，重新计算请见下一例

;例: Fosc = 18.432MHz, Baud0 = 9600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/9600/32/12 + 0.5)
;           = 256 - INT( 5.5 )
;           = 256 - 5
;           = 251
;           = 0FBH
; 2. Baud = 18432000/(256-251)/32/12
;         = 9600
; 3. 一目了然，误差等于零

;例: Fosc = 2.000MHz, Baud = 4800 (1T 模式)
; 1. RELOAD = 256 - INT( 2000000/4800/32 + 0.5)
;           = 256 - INT( 13.02 + 0.5 )
;           = 256 - INT( 13.52 )
;           = 256 - 13
;           = 243
;           = 0F3H
; 2. Baud = 2000000/(256-243)/32
;         = 4808
; 3. error = 0.16%
;------------------------------------------------------------------------------
```

串行通信口一使用独立波特率发生器作串行通信测试程序

```
;/* --- STC International Limited ---------------------------------------------------- */
;/* --- 宏晶科技 姚永平 设计 2007/1/6  V1.0 ------------------------------------------- */
;/* --- 演示 STC11/10xx 系列 MCU 看门狗及其溢出时间计算公式 ---------------------------- */
;/* ---  Mobile: 13922805190 -------------------------------------------------------- */
;/* ---  Fax: 0755-82944243 --------------------------------------------------------- */
;/* ---  Tel: 0755-82948409 --------------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ---------------------------------------------------- */
#include<reg51.h>
#include<intrins.h>
sfr  AUXR     =   0x8e;
sfr  AUXR1    =   0xA2;
sfr  BRT      =   0x9c;
sbit MCU_Start_Led  =  P1^4;
//unsigned char array[9] = {0,2,4,6,8,10,12,14,16};
unsigned char array[9] = {0x00,0x02,0x04,0x06,0x08,0x0A,0x0C,0x0E,0x10};
#define  RELOAD_COUNT  0xfb   //18.432MHz,12T,SMOD=0,9600bps

void serial_port_initial();
void send_UART(unsigned char);
void UART_Interrupt_Receive(void);
void delay(void);
void display_MCU_Start_Led(void);

void main(void)
{
    unsigned  char i = 0;
    serial_port_initial();        //串口初始化
    display_MCU_Start_Led();      //点亮发光二极管表示单片机开始工作
    send_UART(0x34);              //串口发送数据表示单片机串口正常工作
    send_UART(0xa7);              //串口发送数据表示单片机串口正常工作
    for(i = 0;i<9;i++)
    {
        send_UART(array[i]);
    }
    while(1);
}
```

```
/*
void serial_port_initial()      //使用定时器1作为波特率发生器
{
    SCON    =   0x50;           //0101,0000 8位可变波特率，无奇偶校验位
    TMOD    =   0x21;           //0011,0001 设置定时器1为8位自动重装计数器
    TH1     =   RELOAD_COUNT;  //设置定时器1自动重装数
    TL1     =   RELOAD_COUNT;
    TR1     =   1;      //开定时器1
    ES      =   1;      //允许串口中断
    EA      =   1;      //开总中断
}
*/
void serial_port_initial()        //使用独立波特率发生器作为波特率发生器
{
    SCON    =   0x50;   //0101,0000 8位可变波特率，无奇偶校验位
    BRT     =   RELOAD_COUNT;
    AUXR    =   0x11;
                // T0x12,T1x12,UART_M0x6,BRTR,S2SMOD,BRTx12,XRAM,S1BRS
                // Baud = Fosc/(256 - RELOAD_COUNT)/32/12 (12T模式)
                // Baud = Fosc/(256 - RELOAD_COUNT)/32    (1T模式)
                // BRTR = 1,启动独立波特率发生器
                // S1BRS = 1,串口1选择独立波特率发生器作为波特率发生器，
               // 此时定时器1可以释放出来作为定时器，计数器，时钟输出使用
    AUXR1   =   0x80;  //释放该行指令，则串行口从P3口切换到P1口
    ES      =   1;      //允许串口中断
    EA      =   1;      //开总中断
}

void send_UART(unsigned char i)
{
    ES      =   0;  //关串口中断
    TI      =   0;  //清零串口发送完成中断请求标志
    SBUF    =   i;
    while(TI ==0);      //等待发送完成
    TI      =   0;  //清零串口发送完成中断请求标志
    ES      =   1;  //允许串口中断
}
```

```
void UART_Interrupt_Receive(void) interrupt 4
{
     unsigned char   k   =   0;
     if(RI==1)
    {
          RI  =   0;
          k   =   SBUF;
          send_UART(k+1);
    }
    else
    {
          TI  =  0;
    }
}
void delay(void)
{
     unsigned int j  =   0;
     unsigned int g  =   0;
     for(j=0;j<5;j++)
     {
          for(g=0;g<50000;g++)
          {
             _nop_();
             _nop_();
             _nop_();
          }
     }
}
void display_MCU_Start_Led(void)
{
    unsigned  char i = 0;
    for(i=0;i<5;i++)
    {
        MCU_Start_Led    =    0;   //顶亮 MCU 开始工作指示灯
        delay();
        MCU_Start_Led    =    1;   //熄灭 MCU 开始工作指示灯
        delay();
        MCU_Start_Led    =    0;   //顶亮 MCU 开始工作指示灯
    }
}
```

2.23 串行口2的使用

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
S2CON	9Ah	Serial 2 Control register	S2SM0	S2SM1	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0000,0000
S2BUF	9Bh	Serial 2 Data Buffer									xxxx,xxxx
BRT	9Ch	dedicated Baud-Rate timer 独立波特率发生器定时器,装入重装载数									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
IE2	AFh	Interrupt Enable 2							ESPI	ES2	xxxx,xx00
IP2	B5h	2rd Interrupt Priority Low register							PSPI	PS2	xxxx,xx00
IPH2	B6h	2rd Interrupt Priority High register							PSPIH	PS2H	xxxx,xx00
AUXR1	A2h	Auxiliary register 1		PCA_P4	SPI_P4	S2_P4	GF2	ADRJ		DPS	0000,0000
以上是串口2用到的相应寄存器 以下是串口1用到的相应寄存器											
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	BRTR	S2SMOD	BRTx12	EXTRAM	S1BRS	0000,0000
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000

S2SM0	S2SM1	方式	功能说明	串口2波特率
0	0	0	同步移位串行方式	Fosc / 12
0	1	1	8位UART,波特率可变	(2^{S2SMOD} / 32) x BRT独立波特率发生器的溢出率
1	0	2	9位UART	(2^{S2SMOD} / 64) x Fosc系统工作时钟频率
1	1	3	9位UART,波特率可变	(2^{S2SMOD} / 32) x BRT独立波特率发生器的溢出率
BRT独立波特率发生器的溢出率 = Fosc/12/(256 - BRT),当BRTx12 = 0时 BRT独立波特率发生器的溢出率 = Fosc / (256 - BRT),当BRTx12 = 1时				

串口 2 模式 0：

串行数据通过 RxD2/P1.2(RxD2/P4.2)接收,TxD2/P1.3(TxD2/P4.3)输出同步移位时钟，发送接收的是 8 位数据,低位在先，波特率固定在 Fosc / 12，忽略波特率发生器，串口 2 的模式 0 操作和串口 1 的模式 0 操作方式相同。

串口 2 波特率在模式 0 = Fosc 系统工作时钟频率 / 12

串口 2 模式 1：

10 位数据通过 TxD2/P1.3(TxD2/P4.3)发送，通过 RxD2/P1.2(RxD2/P4.2)接收。一祯数据包含一个起始位(0)，8 个数据位，和一个停止位(1)。接收时，停止位进入特殊功能寄存器 S2CON 的 S2RB8 位。波特率由独立波特率发生器 BRT 的溢出率决定。

串口 2 波特率在模式 1 = (2^{S2SMOD} / 32) x BRT 独立波特率发生器的溢出率

当 S2SMOD = 0 时，串口 2 波特率 = BRT 独立波特率发生器的溢出率 / 32，

当 S2SMOD = 1 时，串口 2 波特率 = BRT 独立波特率发生器的溢出率 / 16，

BRT 独立波特率发生器的溢出率 = Fosc/12/(256 - BRT),当 BRTx12 = 0 时，

BRT 独立波特率发生器的溢出率 = Fosc / (256 - BRT),当 BRTx12 = 1 时

串口 2 模式 2：

11 位数据通过 TxD2/P1.3(TxD2/P4.3)发送，通过 RxD2/P1.2(RxD2/P4.2)接收。一祯数据包含一个起始位(0)，8 个数据位，一个可编程的第 9 位，和一个停止位(1)。发送时，第 9 位数据位来自特殊功能寄存器 S2CON 的 S2TB8 位.接收时，第 9 位进入特殊功能寄存器 S2CON 的 S2RB8 位。波特率可编程为系统时钟频率：Fosc / 32 或者 Fosc / 64，串口 2 工作在模式 2 和串口 1 工作在模式 2 是相同的。

串口 2 波特率在模式 2 = (2^{S2SMOD} / 64) x Fosc 系统工作时钟频率

当 S2SMOD = 0 时，串口 2 波特率 = Fosc 系统工作时钟频率 / 64

当 S2SMOD = 1 时，串口 2 波特率 = Fosc 系统工作时钟频率 / 32

串口 2 模式 3：

波特率是可变的，其它和模式 2 相同

11 位数据通过 TxD2/P1.3(TxD2/P4.3)发送，通过 RxD2/P1.2(RxD2/P4.2)接收。一祯数据包含一个起始位(0)，8 个数据位，一个可编程的第 9 位，和一个停止位(1)。发送时，第 9 位数据位来自特殊功能寄存器 S2CON 的 S2TB8 位.接收时，第 9 位进入特殊功能寄存器 S2CON 的 S2RB8 位。

串口 2 波特率在模式 3 = (2^{S2SMOD} / 32) x BRT 独立波特率发生器的溢出率

当 S2SMOD = 0 时，串口 2 波特率 = BRT 独立波特率发生器的溢出率 / 32，

当 S2SMOD = 1 时，串口 2 波特率 = BRT 独立波特率发生器的溢出率 / 16，

BRT 独立波特率发生器的溢出率 = Fosc/12/(256 - BRT),当 BRTx12 = 0 时，

BRT 独立波特率发生器的溢出率 = Fosc / (256 - BRT),当 BRTx12 = 1 时

用户在程序中如何具体使用串口 2

1.设置串口 2 的工作模式，S2CON 寄存器中的 S2SM0 和 S2SM1 两位决定了串口 2 的 4 种工作模式。

2.设置串口 2 的波特率相应的寄存器和位：

BRT 独立波特率发生器寄存器,BRTx12 位，S2SMOD 位

3.启动独立波特率发生器，让 BRTR 位为 1，BRT 独立波特率发生器寄存器就立即开始计数。

4.设置串口 2 的中断优先级，及打开中断相应的控制位是:

PS2,PS2H,ES2,EA

5.如要串口 2 接收，将 S2REN 置 1 即可

如要串口 2 发送，将数据送入 S2BUF 即可，

接收完成标志 S2RI,发送完成标志 S2TI,要由软件清 0。

串行口 2 作串行通信测试程序（C 语言）

```
#include<reg52.h>
#include<intrins.h>
sfr  S2CON    = 0x9A;
//S2SM0,S2SM1,S2SM2,S2REN,S2TB8,SRB8,S2TI,S2RI
sfr  IE2    = 0xAF;
//X,X,X,X,X,X,ESPI,ES2
sfr  S2BUF    = 0x9B;
sfr  AUXR    = 0x8e;
sfr   BRT =    0x9c;
sfr  IAP_CONTR =  0xC7;
sfr  CCON    = 0xD8;
sfr  CMOD    = 0xD9;
sfr  CL      = 0xE9;
sfr  CH      = 0xF9;
sfr  CCAP0L  = 0xEA;
sfr  CCAP0H  = 0xFA;
sfr  CCAPM0  = 0xDA;
sfr  CCAPM1  = 0xDB;
sbit CR      = 0xDE;
sbit MCU_Start_Led  =  P1^7;
sbit S2_Interrupt_Receive_Led  =  P1^4;
//unsigned char self_command_array[4] =  {0x22,0x33,0x44,0x55};
#define Self_Define_ISP_Download_Command  0x22
#define RELOAD_COUNT  0xfb   //18.432MHz,12T,SMOD=0,9600bps
//#define RELOAD_COUNT  0xf6   //18.432MHz,12T,SMOD=0,4800bps
//#define RELOAD_COUNT  0xec   //18.432MHz,12T,SMOD=0,2400bps
//#define RELOAD_COUNT  0xd8   //18.432MHz,12T,SMOD=0,1200bps
void serial_port_one_initial();
void send_UART_one(unsigned char);
void UART_one_Interrupt_Receive(void);

void serial_port_two_initial();
void send_UART_two(unsigned char);
void UART_two_Interrupt_Receive(void);

void soft_reset_to_ISP_Monitor(void);
void delay(void);
void display_MCU_Start_Led(void);
void send_PWM(void);
```

```
void main(void)
{
    unsigned int array_point = 0;
    unsigned char xdata Test_array_one[512]        =
    {
        0x00,   0x01,   0x02,   0x03,   0x04,   0x05,   0x06,   0x07,
        0x08,   0x09,   0x0a,   0x0b,   0x0c,   0x0d,   0x0e,   0x0f,
        0x10,   0x11,   0x12,   0x13,   0x14,   0x15,   0x16,   0x17,
        0x18,   0x19,   0x1a,   0x1b,   0x1c,   0x1d,   0x1e,   0x1f,
        0x20,   0x21,   0x22,   0x23,   0x24,   0x25,   0x26,   0x27,
        0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f,
        0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37,
        0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f,
        0x40,   0x41,   0x42,   0x43,   0x44,   0x45,   0x46,   0x47,
        0x48,   0x49,   0x4a,   0x4b,   0x4c,   0x4d,   0x4e,   0x4f,
        0x50,   0x51,   0x52,   0x53,   0x54,   0x55,   0x56,   0x57,
        0x58,   0x59,   0x5a,   0x5b,   0x5c,   0x5d,   0x5e,   0x5f,
        0x60,   0x61,   0x62,   0x63,   0x64,   0x65,   0x66,   0x67,
        0x68,   0x69,   0x6a,   0x6b,   0x6c,   0x6d,   0x6e,   0x6f,
        0x70,   0x71,   0x72,   0x73,   0x74,   0x75,   0x76,   0x77,
        0x78,   0x79,   0x7a,   0x7b,   0x7c,   0x7d,   0x7e,   0x7f,
        0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87,
        0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f,
        0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97,
        0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f,
        0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7,
        0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf,
        0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7,
        0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf,
        0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7,
        0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf,
        0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7,
        0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf,
        0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7,
        0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef,
        0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7,
        0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff,
        0xff,   0xfe,   0xfd,   0xfc,   0xfb,   0xfa,   0xf9,   0xf8,
        0xf7,   0xf6,   0xf5,   0xf4,   0xf3,   0xf2,   0xf1,   0xf0,
```

```
    0xef,   0xee,   0xed,   0xec,   0xeb,   0xea,   0xe9,   0xe8,
    0xe7,   0xe6,   0xe5,   0xe4,   0xe3,   0xe2,   0xe1,   0xe0,
    0xdf,   0xde,   0xdd,   0xdc,   0xdb,   0xda,   0xd9,   0xd8,
    0xd7,   0xd6,   0xd5,   0xd4,   0xd3,   0xd2,   0xd1,   0xd0,
    0xcf,   0xce,   0xcd,   0xcc,   0xcb,   0xca,   0xc9,   0xc8,
    0xc7,   0xc6,   0xc5,   0xc4,   0xc3,   0xc2,   0xc1,   0xc0,
    0xbf,   0xbe,   0xbd,   0xbc,   0xbb,   0xba,   0xb9,   0xb8,
    0xb7,   0xb6,   0xb5,   0xb4,   0xb3,   0xb2,   0xb1,   0xb0,
    0xaf,   0xae,   0xad,   0xac,   0xab,   0xaa,   0xa9,   0xa8,
    0xa7,   0xa6,   0xa5,   0xa4,   0xa3,   0xa2,   0xa1,   0xa0,
    0x9f,   0x9e,   0x9d,   0x9c,   0x9b,   0x9a,   0x99,   0x98,
    0x97,   0x96,   0x95,   0x94,   0x93,   0x92,   0x91,   0x90,
    0x8f,   0x8e,   0x8d,   0x8c,   0x8b,   0x8a,   0x89,   0x88,
    0x87,   0x86,   0x85,   0x84,   0x83,   0x82,   0x81,   0x80,
    0x7f,   0x7e,   0x7d,   0x7c,   0x7b,   0x7a,   0x79,   0x78,
    0x77,   0x76,   0x75,   0x74,   0x73,   0x72,   0x71,   0x70,
    0x6f,   0x6e,   0x6d,   0x6c,   0x6b,   0x6a,   0x69,   0x68,
    0x67,   0x66,   0x65,   0x64,   0x63,   0x62,   0x61,   0x60,
    0x5f,   0x5e,   0x5d,   0x5c,   0x5b,   0x5a,   0x59,   0x58,
    0x57,   0x56,   0x55,   0x54,   0x53,   0x52,   0x51,   0x50,
    0x4f,   0x4e,   0x4d,   0x4c,   0x4b,   0x4a,   0x49,   0x48,
    0x47,   0x46,   0x45,   0x44,   0x43,   0x42,   0x41,   0x40,
    0x3f,   0x3e,   0x3d,   0x3c,   0x3b,   0x3a,   0x39,   0x38,
    0x37,   0x36,   0x35,   0x34,   0x33,   0x32,   0x31,   0x30,
    0x2f,   0x2e,   0x2d,   0x2c,   0x2b,   0x2a,   0x29,   0x28,
    0x27,   0x26,   0x25,   0x24,   0x23,   0x22,   0x21,   0x20,
    0x1f,   0x1e,   0x1d,   0x1c,   0x1b,   0x1a,   0x19,   0x18,
    0x17,   0x16,   0x15,   0x14,   0x13,   0x12,   0x11,   0x10,
    0x0f,   0x0e,   0x0d,   0x0c,   0x0b,   0x0a,   0x09,   0x08,
    0x07,   0x06,   0x05,   0x04,   0x03,   0x02,   0x01,   0x00
};
    unsigned  char i = 0;

    serial_port_one_initial();        //串口1初始化
    serial_port_two_initial();        //串口2初始化
    display_MCU_Start_Led();    //点亮发光二极管表示单片机开始工作

    send_UART_two(0x55);              //串口2发送数据表示单片机串口正常工作
    send_UART_two(0xaa);              //串口2发送数据表示单片机串口正常工作
```

```
     for(array_point=0; array_point<512; array_point++)
    {
         send_UART_two(Test_array_one[array_point]);
    }

    send_UART_one(0x34);            //串口1发送数据表示单片机串口正常工作
    send_UART_one(0xa7);            //串口1发送数据表示单片机串口正常工作

    for(array_point=0; array_point<512; array_point++)
    {
         send_UART_one(Test_array_one[array_point]);
    }

    send_PWM();                  //6kHz PWM,  50% duty
    while(1);
}
void serial_port_one_initial()
{
    SCON    =   0x50;   //0101,0000 8位可变波特率，无奇偶校验位
//   TMOD    =   0x21;   //0011,0001 设置顶时器1为8位自动重装计数器
//    TH1     =   RELOAD_COUNT;   //设置定时器1自动重装数
//    TL1     =   RELOAD_COUNT;
//    TR1     =   1;     //开定时器1
     BRT  =    RELOAD_COUNT;
//  BRTR = 1, S1BRS = 1, EXTRAM = 1 ENABLE EXTRAM
     AUXR =    0x11; // T0x12,T1x12,UART_M0x6,BRTR,S2SMOD,BRTx12,EXTRAM,S1BRS

    ES      =   1;     //允许串口中断
    EA      =   1;     //开总中断
}
void serial_port_two_initial()
{
//sfr  SCON    = 0x98;
//SM0,SM1,SM2,REN,TB8,RB8,TI,RI

//sfr  S2CON    = 0x9A;
//S2SM0,S2SM1,S2SM2,S2REN,S2TB8,S2RB8,S2TI,S2RI
//sfr  S2BUF    = 0x9B;
//sfr  IE2    = 0xAF;
//X,X,X,X,X,X,ESPI,ES2
```

```
    S2CON    =   0x50;   //0101,0000 8位可变波特率，无奇偶校验位,允许接收

    BRT  =    RELOAD_COUNT;
//  BRTR = 1, S1BRS = 1, EXTRAM = 0 ENABLE EXTRAM
    AUXR =    0x11; // T0x12,T1x12,UART_M0x6,BRTR,S2SMOD,BRTx12,EXTRAM,S1BRS

//     ES      =   1;     //允许串口1中断
//    ES2  =    1
    IE2  =    0x01;     //允许串口2中断,ES2=1
   EA     =   1;     //开总中断
}
void send_UART_one(unsigned char i)
{
   ES     =   0;  //关串口中断
   TI     =   0;  //清零串口发送完成中断请求标志
   SBUF   =   i;
   while(TI ==0); //等待发送完成
   TI     =   0;  //清零串口发送完成中断请求标志
   ES     =   1;  //允许串口中断
}
void send_UART_two(unsigned char i)
{
//sfr  SCON    = 0x98;
//SM0,SM1,SM2,REN,TB8,RB8,TI,RI

//sfr  S2CON    = 0x9A;
//S2SM0,S2SM1,S2SM2,S2REN,S2TB8,S2RB8,S2TI,S2RI
//sfr  S2BUF    = 0x9B;
//sfr  IE2    = 0xAF;
//X,X,X,X,X,X,ESPI,ES2

    unsigned char temp = 0;
//     ES      =   0;  //关串口1中断
    IE2  =    0x00;     //关串口2中断,es2=0
//     TI      =   0;  //清零串口1发送完成中断请求标志
   S2CON =    S2CON & 0xFD; //B'11111101,清零串口2发送完成中断请求标志
//     SBUF   =   i;
   S2BUF   =   i;
//     while(TI ==0); //等待发送完成
```

```
    do
     {
          temp = S2CON;
          temp = temp & 0x02;
     }while(temp==0);


//    TI      =    0;   //清零串口发送完成中断请求标志
    S2CON =     S2CON & 0xFD; //B'11111101,清零串口2发送完成中断请求标志
//     ES       =    1;   //允许串口1中断
//    ES2  =     1
     IE2  =     0x01;      //允许串口2中断,ES2=1
}
void UART_one_Interrupt_Receive(void) interrupt 4
{
    unsigned char    k    =    0;
    if(RI==1)
    {
        RI   =    0;
        k    =    SBUF;
        if(k==Self_Define_ISP_Download_Command)    //是自定义下载命令
        {
            delay();     //延时1秒就足够了
            delay();     //延时1秒就足够了
            soft_reset_to_ISP_Monitor();     //软复位到系统ISP监控区
        }
        send_UART_one(k+1);
    }
    else
    {
        TI   =   0;
    }
}
void UART_two_Interrupt_Receive(void) interrupt 8
{
//sfr   SCON     = 0x98;
//SM0,SM1,SM2,REN,TB8,RB8,TI,RI

//sfr   S2CON     = 0x9A;
//S2SM0,S2SM1,S2SM2,S2REN,S2TB8,S2RB8,S2TI,S2RI
```

```
//sfr  S2BUF    = 0x9B;
//sfr  IE2    = 0xAF;
//X,X,X,X,X,X,ESPI,ES2
     unsigned char  k  =  0;
     k = S2CON ;
     k   = k & 0x01;
     //if(S2RI==1)
    if(k==1)
    {
        //RI  =  0;
         S2CON = S2CON & 0xFE; //1111,1110
         S2_Interrupt_Receive_Led = 0;

        k  =  S2BUF;
        if(k==Self_Define_ISP_Download_Command)  //是自定义下载命令
        {

              delay();    //延时1秒就足够了
            delay();    //延时1秒就足够了

            soft_reset_to_ISP_Monitor();    //软复位到系统ISP监控区
        }
        send_UART_two(k+1);
    }
    else
    {
        //TI  =  0;
         S2CON = S2CON & 0xFD; //1111,1101
    }
}
void soft_reset_to_ISP_Monitor(void)
{
    IAP_CONTR  =   0x60;  //0110,0000  软复位到系统ISP监控区
}
void delay(void)
{
    unsigned int j  =  0;
    unsigned int g  =  0;
    for(j=0;j<5;j++)
    {
```

```
        for(g=0;g<60000;g++)
        {
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
        }
    }
}

void display_MCU_Start_Led(void)
{
//sbit MCU_Start_Led  =  P1^7;
    unsigned  char i = 0;
    for(i=0;i<1;i++)
    {
        MCU_Start_Led   =   0;  //顶亮MCU开始工作指示灯
        delay();
        MCU_Start_Led   =   1;  //熄灭MCU开始工作指示灯
        delay();
        MCU_Start_Led   =   0;  //顶亮MCU开始工作指示灯
    }
}

void send_PWM(void)
{
    CMOD    =   0x00;   // CIDL - - - - CPS1 CPS0 ECF  Setup PCA Timer
                        // CPS1 CPS0 = 00, Fosc/12 is PCA/PWM clock
                        // 18432000/12/256 = 6000
    CL      =   0x00;
    CH      =   0x00;
    CCAP0L  =   0x80;   //Set the initial value same as CCAP0H
    CCAP0H  =   0x80;   //50% Duty Cycle
    CCAPM0  =   0x42;   //0100,0010 Setup PCA module 0 in 8BIT PWM, P3.7
    CR      =   1;      //启动 PCA/PWM 定时器
}
```

串行口 2 作串行通信测试程序（汇编语言）

```
S2CON      EQU     9AH;
;S2SM0,S2SM1,S2SM2,S2REN,S2TB8,SRB8,S2TI,S2RI
IE2  EQU     0AFH
;X,X,X,X,X,X,ESPI,ES2

S2BUF      EQU  9BH

AUXR EQU      8EH
BRT  EQU      9CH
IAP_CONTR    EQU     0C7H

RELOAD_COUNT EQU 0FBH     ;18.432MHz,12T,SMOD=0,9600bps
;RELOAD_COUNT EQU 0F6H    ;18.432MHz,12T,SMOD=0,4800bps
;RELOAD_COUNT EQU 0ECH    ;18.432MHz,12T,SMOD=0,2400bps
;RELOAD_COUNT EQU 0D8H    ;18.432MHz,12T,SMOD=0,1200bps

;=======================================================================
     ORG       0000H
     LJMP MAIN

     ORG  0043H
     LJMP UART_two_Interrupt_Receive

     ORG  0100H
MAIN:
     MOV       SP,  #0C0H
     LCALL     UART2_Initial

     MOV       11H, #55H
     LCALL     send_UART_two

     MOV       11H, #0AAH
     LCALL     send_UART_two

     SJMP $
```

```
;==============================================================================
UART2_Initial:
    PUSH ACC

    MOV     S2CON,    #50H  ;0101,0000 8位可变波特率，无奇偶校验位,允许接收

    MOV     BRT,  #RELOAD_COUNT;

    MOV     AUXR, #11H ;T0x12,T1x12,UART_M0x6,BRTR,S2SMOD,BRTx12,EXTRAM,S1BRS

                    ;BRTR = 1, S1BRS = 1, EXTRAM = 0 ENABLE EXTRAM

    MOV     IE2,  #01H   ;允许串口2中断,ES2=1
    SETB EA         ;开总中断
    POP     ACC
    RET
;==============================================================================
send_UART_two:
    PUSH ACC

//sfr  S2CON    = 0x9A;
//S2SM0,S2SM1,S2SM2,S2REN,S2TB8,S2RB8,S2TI,S2RI
//sfr  S2BUF    = 0x9B;
//sfr  IE2     = 0xAF;
//X,X,X,X,X,X,ESPI,ES2

    MOV     IE2, #00H ;关串口2中断,es2=0
    MOV     A,       S2CON  ;11111101B,清零串口2发送完成中断请求标志
    ANL     A,       #0FDH
    MOV     S2CON,   A

    MOV     S2BUF,   11H
```

```
UART2_SEND_WAIT:
     MOV       A,   S2CON
     ANL       A,   #02H;       0000,0010
     CJNE A,   #02H,     UART2_SEND_WAIT
     MOV       A,   S2CON
     ANL       A,   #0FDH;       1111,1101,清零串口2发送完成中断请求标志
     MOV       S2CON,    A

     MOV       IE2, #01H    ;允许串口2中断,ES2=1
     POP       ACC
     RET
//
  ================================================================================
UART_two_Interrupt_Receive:
     PUSH ACC
     MOV       A,        S2CON
     ANL       A,        #01H
     CJNE A,        #01H,     CLEAR_S2TI_RETI

     MOV       A,        S2CON
     ANL       A,        #0FEH;  1111,1110
     MOV       S2CON,    A

     MOV       11H, S2BUF
     INC       11H
     LCALL     send_UART_two
     POP       ACC
     RETI

CLEAR_S2TI_RETI:
     MOV       A,        S2CON
     ANL       A,        #0FDH;     1111,1101
     MOV       S2CON,    A
     POP       ACC
     RETI

     END
```

2.24 每个单片机具有全球唯一身份证号码(ID号)

宏晶科技最新一代STC12C5201AD/STC12C5A60S2系列每一个单片机出厂时都具有全球唯一身份证号码(ID号)，用户可以在单片机上电后读取内部RAM单元从F1H - F7H连续7个单元的值来获取此单片机的唯一身份证号码(ID号),使用 “ MOV @Ri ” 指令来读取。

2.25 如何知道单片机内部R/C振荡器频率(内部时钟频率)

宏晶科技最新一代STC12C5201AD/STC12C5A60S2系列单片机除了可以使用传统的外部时钟外，还可以选择内部R/C振荡器时钟源(内部时钟).如果选择单片机工作在内部R/C振荡器频率(内部时钟频率),则可以省掉外部晶振。这时XTAL1/XTAL2浮空.但由于使用内部时钟源误差较大,所以在对时序要求较高或者有串行通信的情况下不建议使用内部R/C时钟源。在上电初始化程序时,我们可以通过读取内部RAM单元（FCH,FDH,FEH,FFH连续四个单元)的值来获取单片机出厂时的内部R/C振荡器频率(内部时钟频率)。可以通过读取内部RAM单元（F8H,F9H,FAH,FBH连续四个单元)的值来获取用户最后一次使用内部R/C振荡器时钟下载程序时的频率(内部时钟频率),使用 “ MOV @Ri ” 指令来读取 。

第三章　　STC12系列单片机的I/O口结构

3.1　I/O口各种不同的工作模式及配置介绍

I/O口配置

STC12C5201AD系列单片机其所有I/O口均可由软件配置成4种工作类型之一，如下表所示。4种类型分别为：准双向口（标准8051输出模式）、推挽输出、仅为输入（高阻）或开漏输出功能。每个口由2个控制寄存器中的相应位控制每个引脚工作类型。STC12C5201AD系列单片机上电复位后为准双向口（传统8051的I/O口）模式。2V以上时为高电平，0.8V以下时为低电平。

I/O口工作类型设定

P3口设定 <P3.7, x ,P3.5,P3.4,P3.3,P3.2,P3.1,P3.0　无P3.6口>

P3M1【7：0】	P3M0【7：0】	I/O　口模式
0	0	准双向口(传统8051　I/O　口模式）, 灌电流可达20mA，拉电流为230μA， 由于制造误差，实际为250uA～150uA
0	1	推挽输出（强上拉输出，可达20mA,要加限流电阻）
1	0	仅为输入（高阻）
1	1	开漏(Open　Drain)，内部上拉电阻断开，要外加

P2口设定 <P2.7,P2.6,P2.5,P2.4,P2.3,P2.2,P2.1,P2.0>

P2M1【7：0】	P2M0【7：0】	I/O　口模式
0	0	准双向口(传统8051　I/O　口模式）, 灌电流可达20mA，拉电流为230μA， 由于制造误差，实际为250uA～150uA
0	1	推挽输出（强上拉输出，可达20mA，要加限流电阻）
1	0	仅为输入（高阻）
1	1	开漏(Open　Drain)，内部上拉电阻断开，要外加

P1口设定 <P1.7,P1.6,P1.5,P1.4,P1.3,P1.2,P1.1,P1.0>

P1M1【7：0】	P1M0【7：0】	I/O　口模式（P1.x　如做A/D使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统8051　I/O　口模式）， 灌电流可达20mA，拉电流为230μA， 由于制造误差，实际为250uA～150uA
0	1	推挽输出（强上拉输出，可达20mA，要加限流电阻）
1	0	仅为输入（高阻），如果该I/O口需作为A/D使用，可选此模式
1	1	开漏(Open　Drain)，如果该I/O口需作为A/D使用，可选此模式

P0口设定 < x , x , x , x , P0.3, P0.2, P0.1, P0.0　无P0.7,P0.6,P0.5,P0.4口>

P0M1【7：0】	P0M0【7：0】	I/O　口模式
0	0	准双向口(传统8051　I/O　口模式）, 灌电流可达20mA，拉电流为230μA， 由于制造误差，实际为250uA～150uA
0	1	推挽输出（强上拉输出，可达20mA，要加限流电阻）
1	0	仅为输入（高阻）
1	1	开漏(Open　Drain)，内部上拉电阻断开，要外加

举例：
```
MOV   P1M1,  #10100000B
MOV   P1M0,  #11000000B
;P1.7为开漏,P1.6为强推挽输出,P1.5为高阻输入,P1.4/P1.3/P1.2/P1.1/P1.0为弱上拉
```

注意：

虽然每个I/O口在弱上拉时都能承受20mA的灌电流(还是要加限流电阻,如1K，560Ω等),在强推挽输出时都能输出20mA的拉电流（也要加限流电阻）,但整个芯片的工作电流推荐不要超过55mA。即从MCU-VCC流入的电流不超过55mA,从MCU-Gnd流出电流不超过55mA,整体流入/流出电流都不能超过55mA.

3.2 I/O口各种不同的工作模式结构框图

1.准双向口输出配置

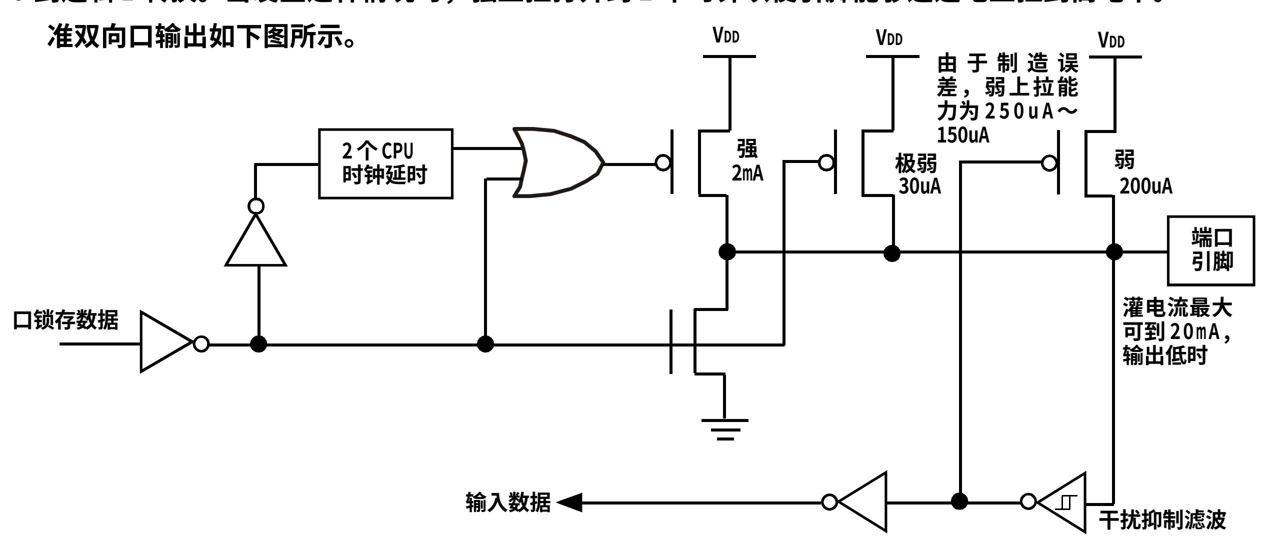
准双向口输出类型可用作输出和输入功能而不需重新配置口线输出状态。这是因为当口线输出为1时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有3个上拉晶体管适应不同的需要。

在3个上拉晶体管中，有1个上拉晶体管称为“弱上拉”，当口线寄存器为1且引脚本身也为1时打开。此上拉提供基本驱动电流使准双向口输出为1。如果一个引脚输出为1而由外部装置下拉到低时，弱上拉关闭而“极弱上拉”维持开状态，为了把这个引脚强拉为低，外部装置必须有足够的灌电流能力使引脚上的电压降到门槛电压以下。

第2个上拉晶体管，称为“极弱上拉”，当口线锁存为1时打开。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流将引脚上拉为高电平。

第3个上拉晶体管称为“强上拉”。当口线锁存器由0到1跳变时，这个上拉用来加快准双向口由逻辑0到逻辑1转换。当发生这种情况时，强上拉打开约2个时钟以使引脚能够迅速地上拉到高电平。

准双向口输出如下图所示。



STC12LE5201AD系列单片机为3V器件，如果用户在引脚加上5V电压，将会有电流从引脚流向VDD，这样导致额外的功率消耗。因此，建议不要在准双向口模式中向3V单片机引脚施加5V电压,如使用的话，要加限流电阻，或用二极管做输入隔离，或用三极管做输出隔离。

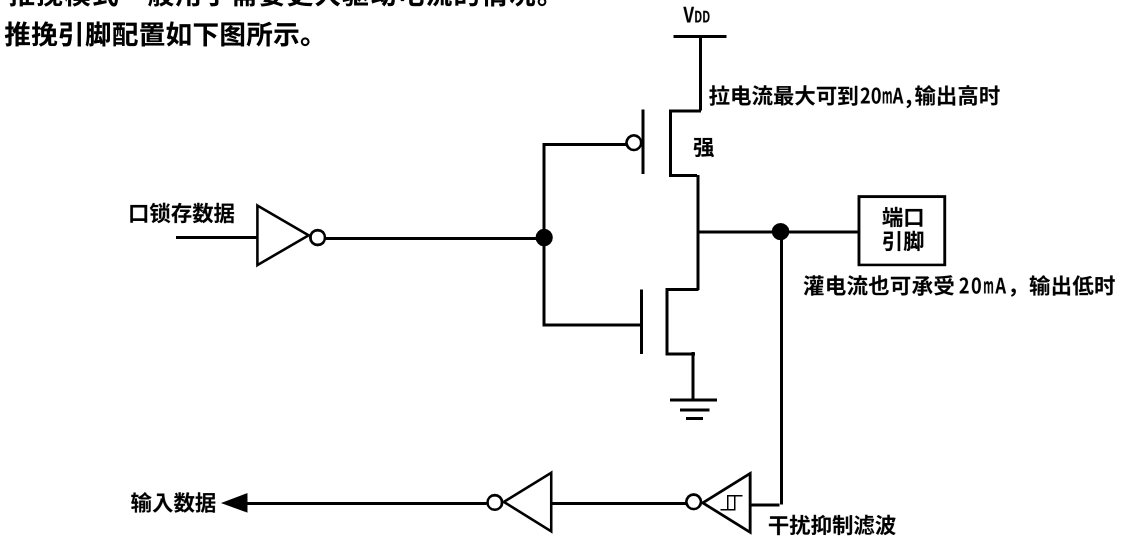
准双向口带有一个施密特触发输入以及一个干扰抑制电路。

准双向口读外部状态前,要先锁存为 ‘1’,才可读到外部正确的状态.

2.推挽输出配置

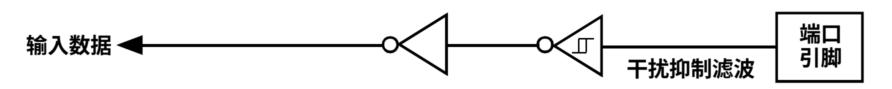
推挽输出配置的下拉结构与开漏输出以及准双向口的下拉结构相同，但当锁存器为1时提供持续的强上拉。推挽模式一般用于需要更大驱动电流的情况。

推挽引脚配置如下图所示。



3．仅为输入（高阻）配置

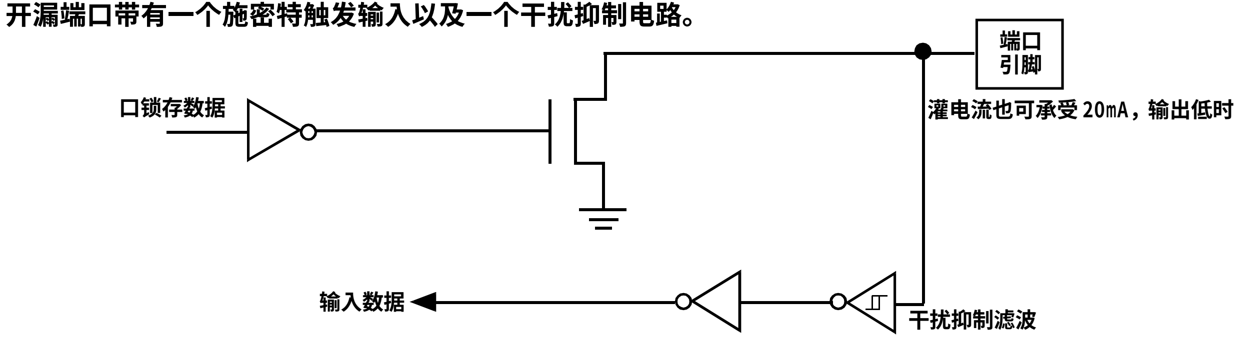
输入口配置如下图所示。



输入口带有一个施密特触发输入以及一个干扰抑制电路。

4.开漏输出配置

当口线锁存器为0时，开漏输出关闭所有上拉晶体管。当作为一个逻辑输出时，这种配置方式必须有外部上拉，一般通过电阻外接到VDD。这种方式的下拉与准双向口相同。输出口线配置如下图所示。
开漏端口带有一个施密特触发输入以及一个干扰抑制电路。



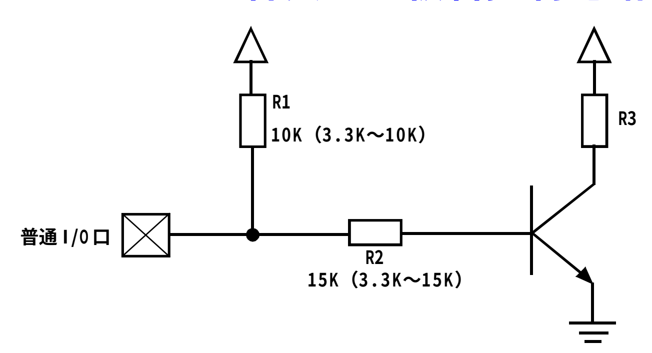
关于I/O口应用注意事项：
少数用户反映I/O口有损坏现象,后发现是

有些是I/O口由低变高读外部状态时,读不对,实际没有损坏,软件处理一下即可
是因为1T的8051单片机速度太快了,软件执行由低变高指令后立即读外部状态,此时由于实际输出还没有变高,就有可能读不对,正确的方法是在软件设置由低变高后加1到2个空操作指令延时,再读就对了.

有些实际没有损坏,加上拉电阻就OK了
是因为外围接的是SPI/I2C等漏极开漏的电路,要加10K上拉电阻.
有些是外围接的是NPN三极管,没有加上拉电阻,其实基极串多大电阻,I/O口就应该上拉多大的电阻,或者将该I/O口设置为强推挽输出.

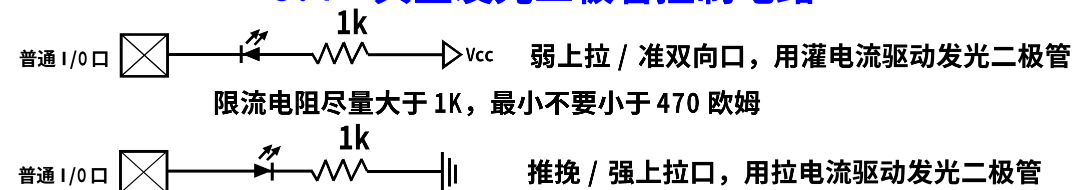
有些确实是损坏了,原因:
发现有些是驱动LED发光二极管没有加限流电阻,建议加1K以上的限流电阻,至少也要加470欧姆以上
发现有些是做行列矩阵按键扫描电路时,实际工作时没有加限流电阻,实际工作时可能出现2个I/O口均输出为低,并且在按键按下时,短接在一起,我们知道一个CMOS电路的2个输出脚不应该直接短接在一起,按键扫描电路中,此时一个口为了读另外一个口的状态,必须先置高才能读另外一个口的状态,而8051单片机的弱上拉口在由0变为1时,会有2个时钟的强推挽高输出电流,输出到另外一个输出为低的I/O口,就有可能造成I/O口损坏.建议在其中的一侧加1K限流电阻,或者在软件处理上,不要出现按键两端的I/O口同时为低.

3.3 一种典型三极管控制电路



如果用弱上拉控制，建议加上拉电阻R1（3.3K～10K），如果不加上拉电阻R1（3.3K～10K），建议R2的值在15K以上，或用强推挽输出。

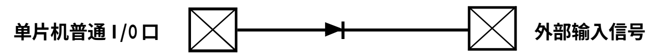
3.4 典型发光二极管控制电路



3.5 3V/5V混合系统I/O口互连

STC12C5201AD系列5V单片机连接3V器件时，为防止3V器件承受不了5V，可将相应的I/O口设置成开漏配置，断开内部上拉电阻，相应的I/O口外部加10K上拉电阻到3V器件的Vcc，这样高电平是3V，低电平是0V，输入输出一切正常。

STC12LE5201AD系列3V单片机连接5V器件时，为防止3V器件承受不了5V，如果相应的I/O口是输入，可在该I/O口上串接一个隔离二极管，隔离高压部分。外部信号电压高于单片机工作电压时截止，I/O口此时已内部上拉到高电平；外部信号电压为低时导通，I/O口被钳位在0.7V，小于0.8V时单片机就认为是低电平。



3.6 如何让I/O口上电复位时为低电平

普通8051单片机上电复位时普通I/O口为弱上拉高电平输出,而很多实际应用要求上电时某些I/O口为低电平输出,否则所控制的系统(如马达)就会误动作,现STC12系列单片机由于既有弱上拉输出又有强推挽输出,就可以很轻松的解决此问题。

现可在STC12系列单片机I/O口上加一个下拉电阻(1K/2K/3K),这样上电复位时，虽然单片机内部I/O口是弱上拉/高电平输出，但由于内部上拉能力有限，而外部下拉电阻又较小,无法将其拉高,所以该I/O口上电复位时外部为低电平。如果要将此I/O口驱动为高电平，可将此I/O口设置为强推挽输出，而强推挽输出时，I/O口驱动电流可达20mA,故肯定可以将该口驱动为高电平输出。

普通I/O口
1K/2K/3K

3.7 PWM输出时I/O口的状态

当某个I/O口作为PWM输出用时,该口的状态:

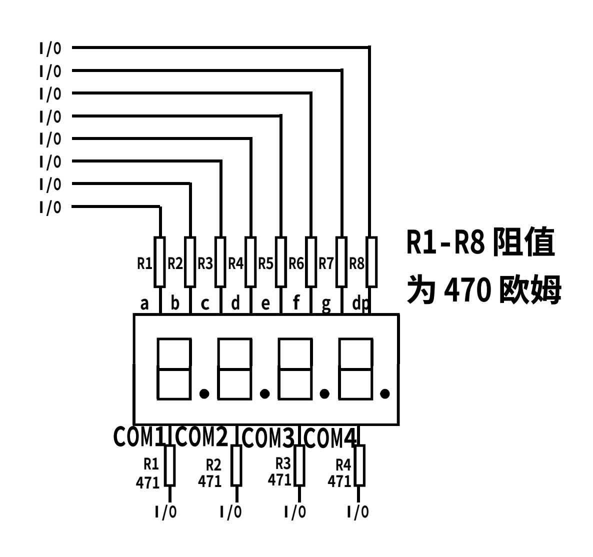
PWM之前口的状态	PWM时口的状态
弱上拉/准双向口	强推挽输出/强上拉输出 要加输出限流电阻10K-1K
强推挽输出	强推挽输出/强上拉输出 要加输出限流电阻10K-1K
仅为输入/高阻	PWM无效
开漏	开漏

限流电阻用10K到1K

普通I/O口 接负载

3.8 I/O 口驱动 LED 数码管应用线路图

Pin	Function	Function	Pin
1	CLKOUT2/P1.0	Vcc	40
2	P1.1	P0.0/AD0	39
3	P1.2	P0.1/AD1	38
4	P1.3	P0.2/AD2	37
5	P1.4	P0.3/AD3	36
6	P1.5	P0.4/AD4	35
7	RxD/INT/P1.6	P0.5/AD5	34
8	TxD/P1.7	P0.6/AD6	33
9	RST/P4.7	P0.7/AD7	32
10	RxD/P3.0	NA/P4.6	31
11	TxD/P3.1	ALE/P4.5	30
12	INT0/P3.2	NA/P4.4	29
13	INT1/P3.3	P2.7/AD15	28
14	CLKOUT0/T0/P3.4	P2.6/AD14	27
15	CLKOUT1/T1/P3.5	P2.5/AD13	26
16	WR/P3.6	P2.4/AD12	25
17	RD/P3.7	P2.3/AD11	24
18	XTAL2	P2.2/AD10	23
19	XTAL1	P2.1/AD9	22
20	Gnd	P2.0/AD8	21



I/O 口动态扫描驱动 4 个
共阴极数码管参考电路图

SOP-32

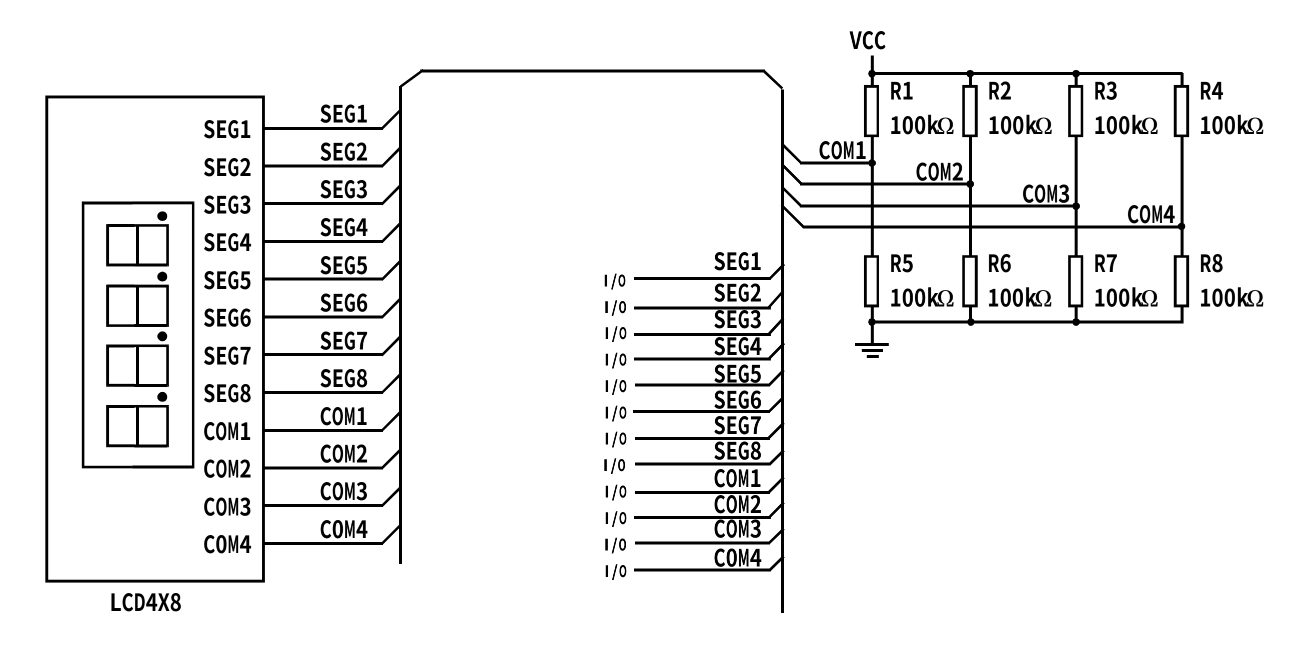
Function	Pin	Pin	Function
P2.2	1	32	VCC
P2.3	2	31	P2.1
RST	3	30	P2.0
RxD/P3.0	4	29	P1.7/ADC7
TxD/P3.1	5	28	P1.6/ADC6
P0.0	6	27	P1.5/ADC5
XTAL2	7	26	P0.3
XTAL1	8	25	P1.4/ADC4
INT0/P3.2	9	24	P1.3/ADC3
P0.1	10	23	P0.2
INT1/P3.3	11	22	P1.2/ADC2/EX_LVD
CLKOUT0/ECI/T0/P3.4	12	21	P1.1/ADC1
CLKOUT1/CCP1/T1/P3.5	13	20	P1.0/ADC0
P2.4	14	19	P3.7/CCP0
P2.5	15	18	P2.7
Gnd	16	17	P2.6

I/O 口动态扫描驱动数码管时，可以一次点亮一个数码管中的 8 段，但为降低功耗，建议可以一次只点亮其中的 4 段或者 2 段

Vcc
LED1 R1 4k7
LED2 R2 4k7
LED3 R3 4k7
LED4 R4 4k7
COM1 COM2 COM3 COM4
a b c d e f g dp
I/O LED1
I/O LED2
I/O LED3
I/O LED4
I/O R5 a
I/O R6 b
I/O R7 c
I/O R8 d
I/O R9 e
I/O R10 f
I/O R11 g
I/O R12 dp
R5-R12 电阻取值 1K

I/O 口动态扫描驱动 4 个
共阳极数码管参考电路图

3.9 I/O 口直接驱动 LCD 应用线路图



如何点亮相应的 LCD 像素:

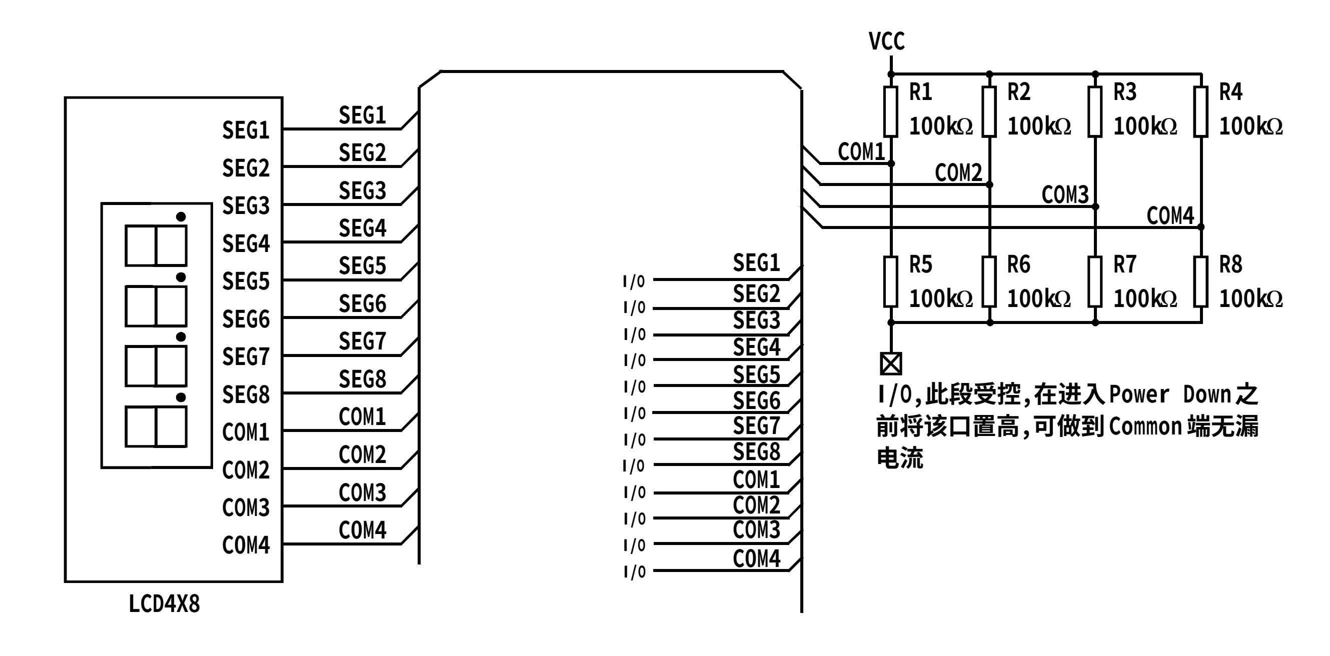
当相应的 Common 端和相应的 Segment 端压差大于 1/2Vcc 时,相应的像素就显示,当压差小于 1/2Vcc 时,相应的像素就不显示

I/O 口如何控制 Segment:

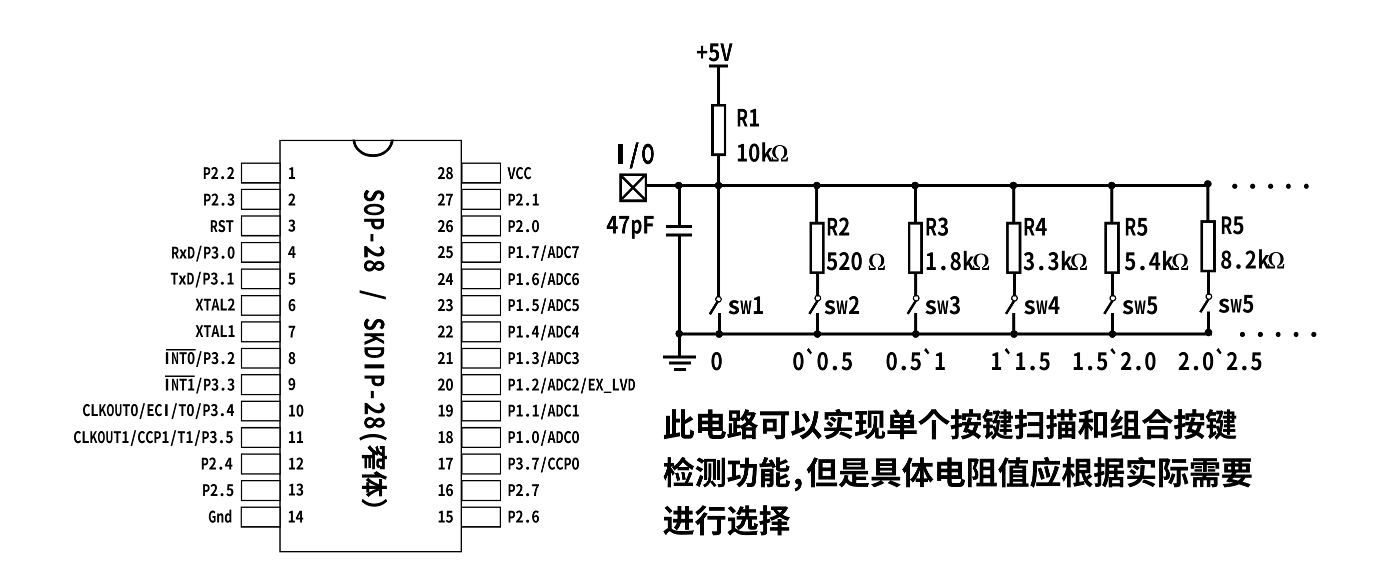
I/O 口直接控制 Segment,程序控制相应的口输出高或低时,对应的 Segment 就是 Vcc 或 0V

I/O 口如何控制 Common:

I/O 口和 2 个 100K 的分压电阻组成 Common,当 I/O 口输出为 0 时,相应的 Common 端为 0V,当 I/O 口强推挽输出为 1 时,相应的 Common 端为 Vcc,当 I/O 口为高阻输入时,相应的 Common 端为 1/2Vcc,

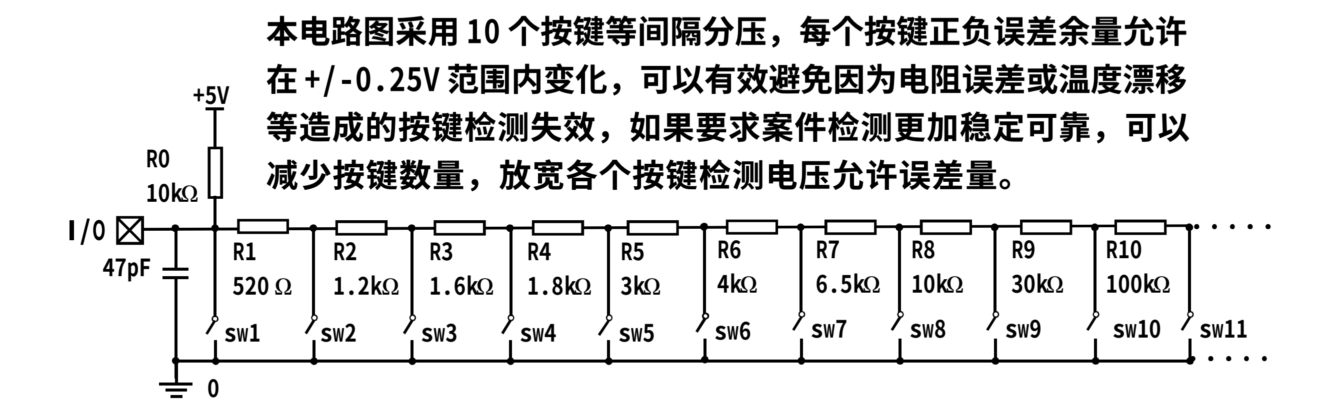


4.0 A/D 做按键扫描应用线路图



此电路可以实现单个按键扫描和组合按键检测功能,但是具体电阻值应根据实际需要进行选择

本电路图采用 10 个按键等间隔分压，每个按键正负误差余量允许在 +/-0.25V 范围内变化，可以有效避免因为电阻误差或温度漂移等造成的按键检测失效，如果要求案件检测更加稳定可靠，可以减少按键数量，放宽各个按键检测电压允许误差量。



第四章 STC12C5201系列单片机看门狗应用及软件复位

4.1 看门狗应用及测试程序

4.1.1 看门狗应用介绍

适用型号： STC12C5201AD 系列

在工业控制 / 汽车电子 / 航空航天等需要高可靠性的系统中,为了防止“系统在异常情况下，受到干扰，MCU/CPU 程序跑飞，导致系统长时间异常工作”,通常是引进看门狗,如果 MCU/CPU 不在规定的时间内按要求访问看门狗,就认为MCU/CPU处于异常状态,看门狗就会强迫MCU/CPU复位,使系统重新从头开始按规律执行用户程序。STC12C5201AD 系列单片机内部也引进了此看门狗功能，使单片机系统可靠性设计变得更加方便 / 简洁。为此功能，我们增加如下特殊功能寄存器 WDT_CONTR：

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	C1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

Symbol 符号 Function 功能

WDT_FLAG When WDT overflows, this bit is set. It can be cleared by software.
看门狗溢出标志位,当溢出时，该位由硬件置 1，可用软件将其清 0。

EN_WDT Enable WDT bit. When set, WDT is started
看门狗允许位， 当设置为“1”时，看门狗启动。

CLR_WDT WDT clear bit. When set, WDT will recount. Hardware will automatically clear this bit.
看门狗清“0”位，当设为“1”时，看门狗将重新计数。硬件将自动清“0”此位。

IDLE_WDT When set, WDT is enabled in IDLE mode. When clear, WDT is disabled in IDLE
看门狗“IDLE”模式位，当设置为“1”时，看门狗定时器在“空闲模式”计数
当清“0”该位时，看门狗定时器在“空闲模式”时不计数

PS2, PS1, PS0 Pre-scale value of Watchdog timer is shown as the bellowed table:
看门狗定时器预分频值，如下表所示

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @20MHz
0	0	0	2	39.3 mS
0	0	1	4	78.6 mS
0	1	0	8	157.3 mS
0	1	1	16	314.6 mS
1	0	0	32	629.1 mS
1	0	1	64	1.25S
1	1	0	128	2.5S
1	1	1	256	5S

The WDT period is determined by the following equation 看门狗溢出时间计算
看门狗溢出时间 = (12 x Pre-scale x 32768) / Oscillator frequency
设时钟为 12MHz：
看门狗溢出时间 = (12 x Pre-scale x 32768) / 12000000 = Pre-scale x 393216 / 12000000

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @12MHz
0	0	0	2	65.5 mS
0	0	1	4	131.0 mS
0	1	0	8	262.1 mS
0	1	1	16	524.2 mS
1	0	0	32	1.0485S
1	0	1	64	2.0971S
1	1	0	128	4.1943S
1	1	1	256	8.3886S

设时钟为 11.0592MHz：

看门狗溢出时间 = (12 x Pre-scale x 32768) / 11059200 = Pre-scale x 393216 / 11059200

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @11.0592MHz
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377S
1	0	1	64	2.2755S
1	1	0	128	4.5511S
1	1	1	256	9.1022S

汇编语言程序示例

```
WDT_CONTR    DATA0C1H ;    或者    WDT_CONTR    EQU    0C1H
;复位入口
    ORG      0000H
    LJMPInitial
    …
    ORG      0060H
Initial:
    MOV      WDT_CONTR,   #00111100B; Load initial value 看门狗定时器控制寄存器初始化
                ; EN_WDT = 1, CLR_WDT = 1, IDLE_WDT = 1, PS2 = 1, PS1 = 0, PS0 = 0
    …
Main_Loop:
    LCALL    Display_Loop
    LCALL    Keyboard_Loop
    …
    MOV      WDT_CONTR,   #00111100B ;  喂狗, 不要用 ORL    WDT_CONTR,   #00010000B
    …
    LJMPMain_Loop
```

C 语言程序示例

```
#include<reg52.h>
sfr      WDT_CONTR    =    0xc1;
void main()
{    …
    WDT_CONTR    =    0x3c;
       /* 0011,1100 EN_WDT = 1,CLR_WDT = 1,IDLE_WDT = 1,PS2 = 1,PS1 = 0,PS0 = 0 */
    while(1){
        display();
        keyboard();
        …
        WDT_CONTR    =    0x3c;  /* 喂狗, 不要用 WDT_CONTR    =    WDT_CONTR | 0x10;*/
    }
}
```

4.1.2 一个完整的看门狗测试程序，在宏晶的下载板上可以直接测试

本程序验证 STC12C5201AD 系列单片机的看门狗及其溢出时间计算公式

```
;/* ---  STC International Limited -------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 --------------- */
;/* ---  演示 STC12C5201AD 系列 MCU 看门狗及其溢出时间计算公式 - */
;/* ---  Mobile: 13922805190 -------------------------------- */
;/* ---  Fax: 0755-82944243 --------------------------------- */
;/* ---  Tel: 0755-82948409 --------------------------------- */
;/* ---  Web: www.mcu-memory.com ---------------------------- */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过,相关的工作状态在 P1 口上显示
;看门狗及其溢出时间 = (12 * Pre_scale *32768)/Oscillator frequency
WDT_CONTR        EQU 0C1H ;看门狗地址
WDT_TIME_LED     EQU P1.5 ;用 P1.5 控制看门狗溢出时间指示灯,
                          ;看门狗溢出时间可由该指示灯亮的时间长度或熄灭的时间长度表示
WDT_FLAG_LED     EQU P1.7 ;用 P1.7 控制看门狗溢出复位指示灯，如点亮表示为看门狗溢出复位
Last_WDT_Time_LED_Status   EQU 00H ;位变量，存储看门狗溢出时间指示灯的上一次状态位
;WDT 复位时间(所用的Oscillator frequency = 18.432MHz):
;Pre_scale_Word  EQU 00111100B   ;清0,启动看门狗，预分频数=32，   0.68S
Pre_scale_Word  EQU 00111101B    ;清0,启动看门狗，预分频数=64，   1.36S
;Pre_scale_Word  EQU 00111110B   ;清0,启动看门狗，预分频数=128，  2.72S
;Pre_scale_Word  EQU 00111111B   ;清0,启动看门狗，预分频数=256，  5.44S
    ORG   0000H
    AJMP  MAIN
    ORG   0100H
MAIN:
    MOV   A, WDT_CONTR           ;检测是否为看门狗复位
    ANL   A, #10000000B
    JNZ   WDT_Reset              ;WDT_CONTR.7 = 1，看门狗复位，跳转到看门狗复位程序
;WDT_CONTR.7 = 0,上电复位，冷启动，RAM 单元内容为随机值
    SETB  Last_WDT_Time_LED_Status        ;上电复位，
                                          ;初始化看门狗溢出时间指示灯的状态位 = 1
    CLR   WDT_TIME_LED                    ;上电复位，点亮看门狗溢出时间指示灯
    MOV   WDT_CONTR, #Pre_scale_Word      ;启动看门狗
WAIT1:
    SJMP  WAIT1                   ;循环执行本语句(停机)，等待看门狗溢出复位

;WDT_CONTR.7 = 1,看门狗复位，热启动，RAM 单元内容不变，为复位前的值
WDT_Reset:                       ;看门狗复位，热启动
    CLR   WDT_FLAG_LED           ;是看门狗复位,点亮看门狗溢出复位指示灯

    JB   Last_WDT_Time_LED_Status, Power_Off_WDT_TIME_LED;为1熄灭相应的灯,为0亮相应灯
    ;根据看门狗溢出时间指示灯的上一次状态位设置 WDT_TIME_LED 灯,
    ;若上次亮本次就熄灭，若上次熄灭本次就亮
```

```
    CLR    WDT_TIME_LED                 ;上次熄灭本次点亮看门狗溢出时间指示灯
    CPL    Last_WDT_Time_LED_Status     ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT2:
    SJMP   WAIT2                        ;循环执行本语句(停机)，等待看门狗溢出复位
Power_Off_WDT_TIME_LED:
    SETB   WDT_TIME_LED                 ;上次亮本次就熄灭看门狗溢出时间指示灯
    CPL    Last_WDT_Time_LED_Status     ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT3:
    SJMP   WAIT3                        ;循环执行本语句(停机)，等待看门狗溢出复位
    END
```

4.2 如何用软件实现系统复位

用户应用程序在运行过程当中，有时会有特殊需求，需要实现单片机系统软复位（热启动之一），传统的8051单片机由于硬件上未支持此功能，用户必须用软件模拟实现，实现起来较麻烦。现STC新推出的增强型8051根据客户要求增加了IAP_CONTR 特殊功能寄存器,实现了此功能。用户只需简单的控制IAP_CONTR 特殊功能寄存器的其中两位 SWBS / SWRST 就可以系统复位了。

IAP_CONTR: IAP控制寄存器，地址在 0C7H 单元

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
IAPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0	0000,x000

IAPEN: ISP/IAP 功能允许位。0：禁止IAP编程改变Flash,1:允许编程改变Flash
SWBS: 软件选择从用户应用程序区启动（0），还是从ISP程序区启动（1）。要与SWRST直接配合才可以实现
SWRST: 0: 不操作； 1: 产生软件系统复位，硬件自动清零。
CMD_FAIL: 如果送了ISP/IAP命令，并对IAP_TRIG送5Ah/A5h触发失败，则为1，需由软件清零。

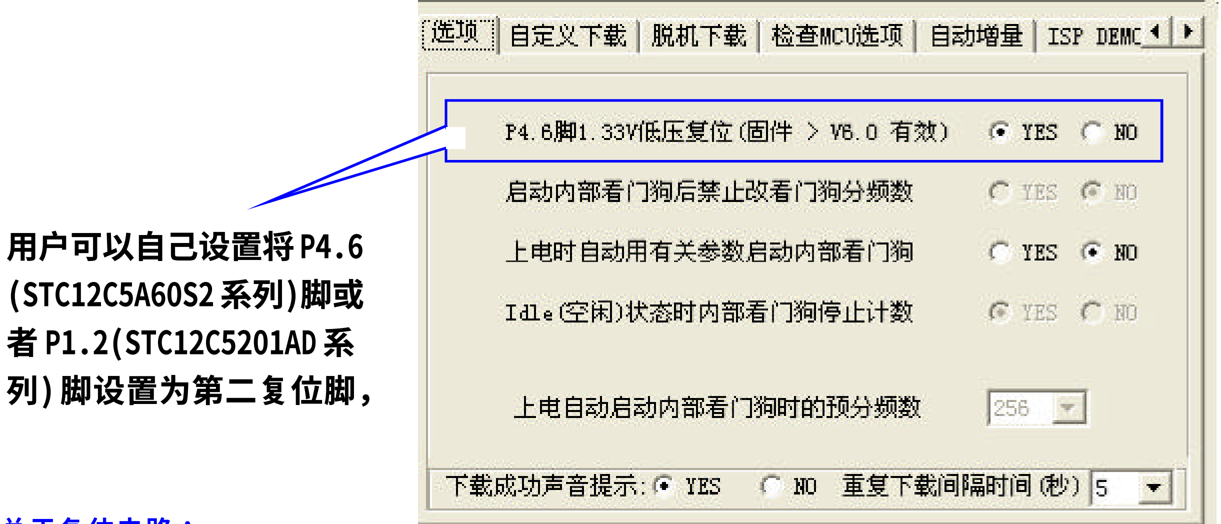
```
;从用户应用程序区(AP区)软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV    IAP_CONTR, #00100000B  ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;从系统ISP监控程序区软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV    IAP_CONTR, #00100000B  ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;从用户应用程序区(AP区)软件复位并切换到系统ISP监控程序区开始执行程序
    MOV    IAP_CONTR, #01100000B  ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
;从系统ISP监控程序区软件复位并切换到系统ISP监控程序区开始执行程序
    MOV    IAP_CONTR, #01100000B  ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
```

本复位是整个系统复位，所有的特殊功能寄存器都会复位到初始值，I/O口也会初始化。

4.3 热启动复位和冷启动复位

	复位源	现象
热启动复位	内部看门狗复位	会使单片机直接从用户程序区0000H处开始执行用户程序
	通过控制RESET脚产生的硬复位	会使系统 从用户程序区0000H处开始直接执行用户程序
	通过对IAP_CONTR寄存器送入20H产生的软复位	会使系统从用户程序区0000H处开始直接执行用户程序
	通过对IAP_CONTR寄存器送入60H产生的软复位	会使系统从系统ISP监控程序区开始执行程序，检测不到合法的ISP下载命令流后，会软复位到用户程序区执行用户程序
冷启动复位	系统停电后再上电引起的硬复位	会使系统从系统ISP监控程序区开始执行程序，检测不到合法的ISP下载命令流后，会软复位到用户程序区执行用户程序

4.4 新增第二复位功能脚选择与应用

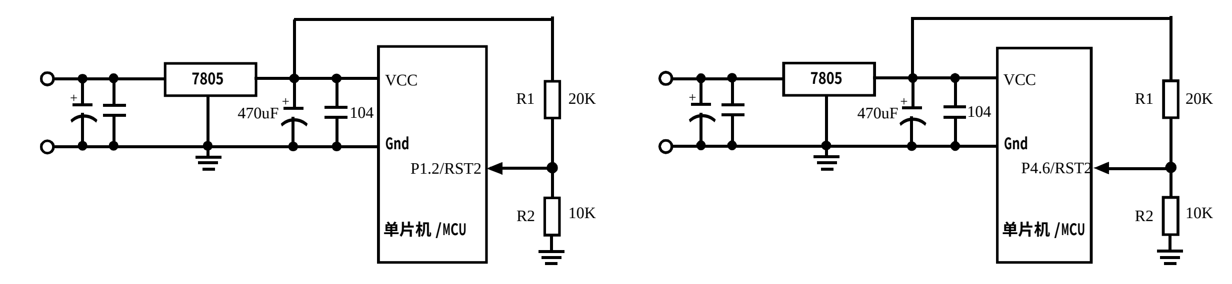


用户可以自己设置将P4.6(STC12C5A60S2系列)脚或者P1.2(STC12C5201AD系列)脚设置为第二复位脚,

关于复位电路:

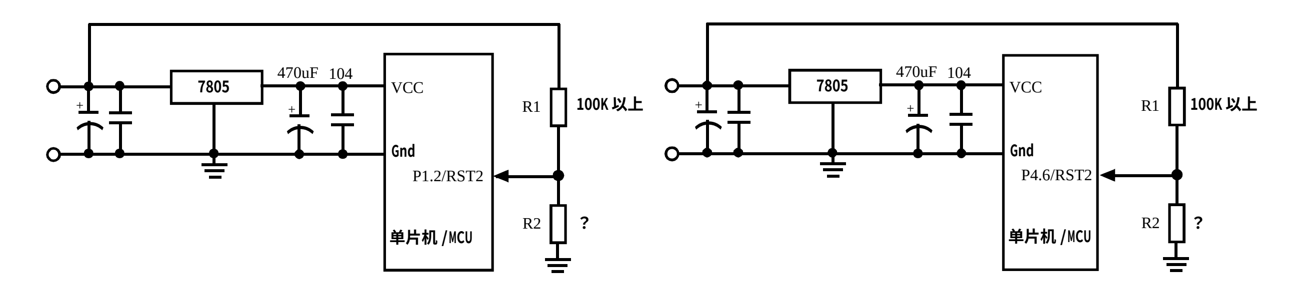
时钟频率高于12MHz时,建议使用第二复位功能脚(STC12C5A60S2系列在RST2/EX_LVD/P4.6口 STC12C5201AD系列在RST2/EX_LVD/P1.2口)

利用增加的外部低压检测LVD功能作外部低压检测复位脚,典型应用线路图



STC12C5201AD系列外部低压检测LVD在P1.2口 **STC12C5A60S2系列外部低压检测LVD在P4.6口**

上图中,稳压块7805后端的直流电是5V,稳压块7805后端的直流电电掉到4V附近时,上图中的电阻R1和R2将4V附近的电压分压到低于低压检测门槛电压(1.33V附近)。此时第二复位功能脚RST2就让CPU处于复位状态,当稳压块7805后端的直流电压高于4V以上时,上图中的电阻R1和R2将4V的电压分压到高于低压检测门槛电压(1.33V附近),单片机就解除复位状态,恢复到正常工作状态.



如交流电在220V时,稳压块7805前端的直流电是11V,当交流电降到160V时,稳压块7805前端的直流电是8.5V,上图中的电阻R1和R2将8.5V的电压分压到低于低压检测门槛电压(1.33V附近)。此时第二复位功能脚RST2就让CPU处于复位状态,当稳压块7805前端的直流电压高于8.5V以上时,上图中的电阻R1和R2将8.5V的电压分压到高于低压检测门槛电压(1.33V附近),单片机就解除复位状态,恢复到正常工作状态.

第五章 STC12 系列单片机 EEPROM 的应用

--- 利用 ISP/IAP 技术将内部 Data Flash 当 EEPROM，擦写次数 10 万次以上

5.1 IAP 及 EEPROM 新增特殊功能寄存器介绍

5V 单片机在 3.7V 以上对 EEPROM 进行操作才有效,3.7V 以下对 EEPROM 进行操作,MCU 不执行此功能,但会继续往下执行程序.3.3V 单片机在 2.4V 以上对 EEPROM 进行操作才有效,2.4V 以下对 EEPROM 进行操作,MCU 不执行此功能,但会继续往下执行程序.所以建议上电复位后在初始化程序时加 200MS 延时.

STC12 系列 1T 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IAP_DATA	C2h	ISP/IAP Flash Data Register									1111,1111
IAP_ADDRH	C3h	ISP/IAP Flash Address High									0000,0000
IAP_ADDRL	C4h	ISP/IAP Flash Address Low									0000,0000
IAP_CMD	C5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,xx00
IAP_TRIG	C6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
IAP_CONTR	C7h	ISP/IAP Control Register	IAPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0	0000,x000

IAP_DATA: ISP/IAP 操作时的数据寄存器。
ISP/IAP 从 Flash 读出的数据放在此处，向 Flash 写的数据也需放在此处

IAP_ADDRH: ISP/IAP 操作时的地址寄存器高八位。

IAP_ADDRL: ISP/IAP 操作时的地址寄存器低八位。

IAP_CMD: ISP/IAP 操作时的命令模式寄存器，须命令触发寄存器触发方可生效。

B7	B6	B5	B4	B3	B2	B1	B0	命令 / 操作 模式选择
保留						命令		
-	-	-	-	-	-	0	0	Standby 待机模式，无ISP操作
-	-	-	-	-	-	0	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节读
-	-	-	-	-	-	1	0	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节编程
-	-	-	-	-	-	1	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行扇区擦除

程序在用户应用程序区时，仅可以对数据Flash区(EEPROM)进行字节读/字节编程/扇区擦除,STC12C5206AD/STC12C5206PWM/12LE5206AD/12LE5206PWM 除外，这几个型号可在应用程序区修改应用程序区。

IAP_TRIG: ISP/IAP 操作时的命令触发寄存器。
在 IAPEN(IAP_CONTR.7) = 1 时,对 IAP_TRIG 先写入 5Ah,再写入 A5h,ISP/IAP 命令才会生效。

IAP_CONTR: ISP/IAP 控制寄存器，地址在 0C7H 单元

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
IAPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

IAPEN: ISP/IAP 功能允许位。0：禁止 ISP/IAP 编程改变 Flash,1:允许编程改变 Flash

SWBS: 软件选择从用户主程序区启动（0），还是从 ISP 程序区启动（1）。

SWRST: 0: 不操作； 1: 产生软件系统复位，硬件自动清零。

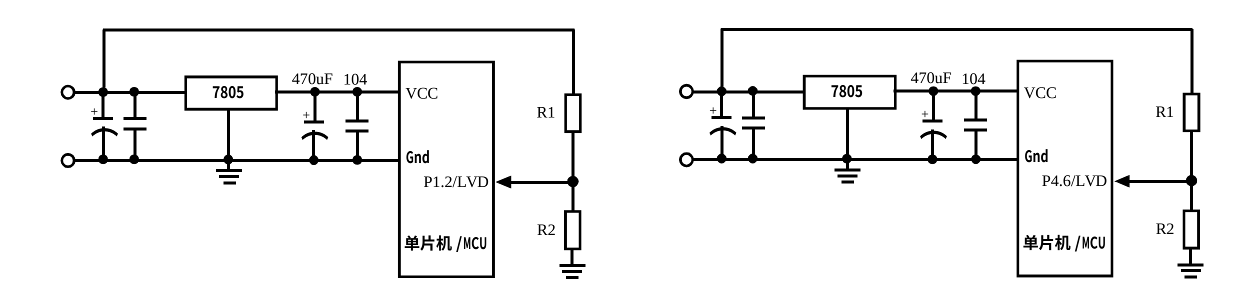
CMD_FAIL: 如果送了 ISP/IAP 命令，并对 ISP_TRIG 送 5Ah/A5h 触发失败，则为 1，需由软件清零。

```
;在用户应用程序区(AP 区)软件复位并从用户应用程序区(AP 区)开始执行程序
      MOV    IAP_CONTR, #00100000B  ;SWBS = 0(选择 AP 区), SWRST = 1(软复位)
;在用户应用程序区(AP 区)软件复位并从系统 ISP 监控程序区开始执行程序
      MOV    IAP_CONTR, #01100000B  ;SWBS = 1(选择 ISP 区), SWRST = 1(软复位)
;在系统 ISP 监控程序区软件复位并从用户应用程序区(AP 区)开始执行程序
      MOV    IAP_CONTR, #00100000B  ;SWBS = 0(选择 AP 区), SWRST = 1(软复位)
;在系统 ISP 监控程序区软件复位并从系统 ISP 监控程序区开始执行程序

MOV   IAP_CONTR, #01100000B  ;SWBS = 1(选择 ISP 区), SWRST = 1(软复位)
```

设置等待时间			CPU 等待时间(多少个 CPU 工作时钟)			
WT2	WT1	WT0	Read/读	Program/编程	Sector Erase 扇区擦除	Recommended System Clock 跟等待参数对应的推荐系统时钟
1	1	1	2个时钟	55个时钟	21012个时钟	〈 1MHz
1	1	0	2个时钟	110个时钟	42024个时钟	〈 2MHz
1	0	1	2个时钟	165个时钟	63036个时钟	〈 3MHz
1	0	0	2个时钟	330个时钟	126072个时钟	〈 6MHz
0	1	1	2个时钟	660个时钟	252144个时钟	〈 12MHz
0	1	0	2个时钟	1100个时钟	420240个时钟	〈 20MHz
0	0	1	2个时钟	1320个时钟	504288个时钟	〈 24MHz
0	0	0	2个时钟	1760个时钟	672384个时钟	〈 30MHz

利用增加的外部低压检测 LVD 功能作外部低压检测，判断是否要开始保存数据典型应用线路图



如交流电在 220V 时，稳压块 7805 前端的直流电是 11V,当交流电降到 160V 时，稳压块 7805 前端的直流电是 8.5V，图中的电阻 R1 和 R2 将 8.5V 的电压分压到低于低压检测门槛电压。此时 CPU 可以用查询方式查询，推荐使用中断，在中断服务程序里面，将 LVDF 位清零，再读 LVDF 位。如果为 0，则认为是电源抖动，如果为 1，则认为电源掉电，立即进行保存现场数据的工作。保存现场完成后，再将 LVDF 位清零，再读 LVDF 位的值。如果为 0，则认为电源系统恢复正常，此时 CPU 可恢复正常工作，如果为 1，继续将 LVDF 位清 0，再读 LVDF 的值，用此方法，等到电源恢复正常，或电源彻底掉电，CPU 进入复位状态。

5.2　STC12C5201AD系列单片机EEPROM地址

STC12C5201AD系列单片机内部可用Data Flash(EEPROM)的地址(与程序空间是分开的)：如果对应用程序区进行IAP写数据/擦除扇区的动作，则该语句会被单片机忽略，继续执行下一句。程序在用户应用程序区(AP区)时,仅可以对Data Flash(EEPROM)进行IAP/ISP操作。

STC12C5206AD/12C5206PWM/12LE5206AD/12LE5206PWM除外，这几个型号可在应用程序区修改应用程序区。

STC12C5201AD系列单片机的内部EEPROM地址表				
第一扇区		第二扇区		每个扇区 512字节,共2个扇区 建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不必用满，当然可全用，用满则为2K字节EEPROM。由于擦除是按扇区擦除，所以每个扇区用的越少越方便，256个字节以内较合理。
起始地址	结束地址	起始地址	结束地址	
0000h	01FFh	0200h	03FFh	
适用的型号如下： STC12C5201,STC12C5201AD，STC12C5201PWM，STC12LE5201,STC12LE5201AD,STC12LE5201PWM STC12C5202,STC12C5202AD，STC12C5202PWM，STC12LE5202,STC12LE5202AD,STC12LE5202PWM STC12C5204,STC12C5204AD，STC12C5204PWM，STC12LE5204,STC12LE5204AD,STC12LE5204PWM STC12C5205,STC12C5205AD，STC12C5205PWM，STC12LE5205,STC12LE5205AD,STC12LE5205PWM				

STC12C5206AD/12LE5206AD/12C5206PWM/12LE5206PWM,这几个型号可在应用程序区修改应用程序区。单片机可对自身内部应用程序区进行IAP操作，故所有部分均可当Data Flash(EEPROM)使用，其扇区地址如下，最大空间6K：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不必用满，当然可全用
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0000h	01FFh	0200h	03FFh	0400h	05FFh	0600h	07FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0800h	09FFh	0A00h	0BFFh	0C00h	0DFFh	0E00h	0FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1000h	11FFh	1200h	13FFh	1400h	15FFh	1600h	17FFh	

STC12C5A60S2/AD/PWM系列单片机的EEPROM起始地址从0000H开始，每个扇区512字节，类推下去从0200H开始。

5.3　STC12C5A60S2系列单片机EEPROM地址

STC12C5A32S2/AD/PWM/CCP单片机的内部EEPROM地址表 STC12LE5A32S2/AD/PWM/CCP单片机的内部EEPROM地址表								
第一扇区		第二扇区		第三扇区		第四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0000h	1FFh	200h	3FFh	400h	5FFh	600h	7FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
800h	9FFh	A00h	BFFh	C00h	DFFh	E00h	FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1000h	11FFh	1200h	13FFh	1400h	15FFh	1600h	17FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1800h	19FFh	1A00h	1BFFh	1C00h	1DFFh	1E00h	1FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	每个扇区512字节 建议同一次修改的数据放在同一扇区
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFH	
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
3000h	31FFh	3200h	33FFh	3400h	35FFh	3600h	37FFH	
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
3800h	39FFh	3A00h	3BFFh	3C00h	3DFFh	3E00h	3FFFH	
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
4000h	41FFh	4200h	43FFh	4400h	45FFh	4600h	47FFH	
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
4800h	49FFh	4A00h	4BFFh	4C00h	4DFFh	4E00h	4FFFH	
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
5000h	51FFh	5200h	53FFh	5400h	55FFh	5600h	57FFH	
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
5800h	59FFh	5A00h	5BFFh	5C00h	5DFFh	5E00h	5FFFH	
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
6000h	61FFh	6200h	63FFh	6400h	65FFh	6600h	67FFH	
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
6800h	69FFh	6A00h	6BFFh	6C00h	6DFFh	6E00h	6FFFH	

STC12C5A60S2/AD/CCP系列单片机内部EEPROM选型一览表 STC12LE5A60S2/AD/CCP系列单片机内部EEPROM选型一览表				
型号	EEPROM字节数	扇区数	起始扇区首地址	结束扇区末尾地址
STC12C5A08S2/AD/PWM	8K	16	0000h	1FFFh
STC12C5A16S2/AD/PWM	8K	16	0000h	1FFFh
STC12C5A20S2/AD/PWM	8K	16	0000h	1FFFh
STC12C5A32S2/AD/PWM	28K	56	0000h	6FFFh
STC12C5A40S2/AD/PWM	20K	40	0000h	4FFFh
STC12C5A48S2/AD/PWM	12K	24	0000h	2FFFh
STC12C5A52S2/AD/PWM	8K	16	0000h	1FFFh
STC12C5A56S2/AD/PWM	4K	8	0000h	0FFFh
STC12C560S2/AD/PWM	1K	2	0000h	03FFh
STC12LE5A08S2/AD/PWM	8K	16	0000h	1FFFh
STC12LE5A16S2/AD/PWM	8K	16	0000h	1FFFh
STC12LE5A20S2/AD/PWM	8K	16	0000h	1FFFh
STC12LE5A32S2/AD/PWM	28K	56	0000h	6FFFh
STC12LE5A40S2/AD/PWM	20K	40	0000h	4FFFh
STC12LE5A48S2/AD/PWM	12K	24	0000h	2FFFh
STC12LE5A52S2/AD/PWM	8K	16	0000h	1FFFh
STC12LE5A56S2/AD/PWM	4K	8	0000h	0FFFh
STC12LE560S2/AD/PWM	1K	2	0000h	03FFh

5.4 IAP及EEPROM汇编简介

```
;用 DATA 还是 EQU 声明新增特殊功能寄存器地址要看你用的汇编器 / 编译器
    IAP_DATA         DATA      0C2h; 或   IAP_DATA     EQU     0C2h
    IAP_ADDRH        DATA      0C3h; 或   IAP_ADDRH    EQU     0C3h
    IAP_ADDRL        DATA      0C4h; 或   IAP_ADDRL    EQU     0C4h
    IAP_CMD          DATA      0C5h; 或   IAP_CMD      EQU     0C5h
    IAP_TRIG         DATA      0C6h; 或   IAP_TRIG     EQU     0C6h
    IAP_CONTR        DATA      0C7h; 或   IAP_CONTR    EQU     0C7h
;定义 ISP/IAP 命令及等待时间
    ISP_IAP_BYTE_READ        EQU   1     ;字节读
    ISP_IAP_BYTE_PROGRAM     EQU   2     ;字节编程,前提是该字节是空，0FFh
    ISP_IAP_SECTOR_ERASE     EQU   3     ;扇区擦除,要某字节为空，要擦一扇区
    WAIT_TIME         EQU   0            ;设置等待时间,30MHz 以下 0,24M 以下 1,
          ;20MHz 以下 2,12M 以下 3,6M 以下 4,3M 以下 5,2M 以下 6,1M 以下 7,
;字节读
    MOV   IAP_ADDRH, #BYTE_ADDR_HIGH      ;送地址高字节  ┐
    MOV   IAP_ADDRL, #BYTE_ADDR_LOW       ;送地址低字节  ┘ 地址需要改变时才需重新送地址
    MOV   IAP_CONTR, #WAIT_TIME  ;设置等待时间      ┐
    ORL   IAP_CONTR, #10000000B  ;允许 ISP/IAP 操作 ┘ 此两句可以合成一句,并且只送一次就够了
    MOV   IAP_CMD,        #ISP_IAP_BYTE_READ;送字节读命令,命令不需改变时,不需重新送命令
    MOV   IAP_TRIG,       #5Ah        ;先送 5Ah,再送 A5h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   IAP_TRIG,       #0A5h       ;送完 A5h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序。
    NOP                               ;数据读出到 IAP_DATA 寄存器后，CPU 继续执行程序
    MOV   A,         ISP_DATA         ;将读出的数据送往 Acc
;以下语句可不用,只是出于安全考虑而已
    MOV   IAP_CONTR, #00000000B       ;禁止 ISP/IAP 操作
    MOV   IAP_CMD,   #00000000B       ;去除 ISP/IAP 命令
    ;MOV  IAP_TRIG,  #00000000B       ;防止 ISP/IAP 命令误触发
    ;MOV  IAP_ADDRH, #0FFh            ;送地址高字节单元为 00,指向非 EEPROM 区
    ;MOV  IAP_ADDRL, #0FFh            ;送地址低字节单元为 00,防止误操作
```

```
;字节编程，该字节为FFh/空时，可对其编程，否则不行,要先执行扇区擦除
    MOV   IAP_DATA,  #ONE_DATA     ;送字节编程数据到IAP_DATA,只有数据改变时才需重新送
    MOV   IAP_ADDRH, #BYTE_ADDR_HIGH  ;送地址高字节  ┐
    MOV   IAP_ADDRL, #BYTE_ADDR_LOW   ;送地址低字节  ┘ 地址需要改变时才需重新送地址
    MOV   IAP_CONTR, #WAIT_TIME  ;设置等待时间     ┐
    ORL   IAP_CONTR, #10000000B  ;允许ISP/IAP操作  ┘ 此两句可合成一句,并且只送一次就够了
    MOV   IAP_CMD,   #ISP_IAP_BYTE_PROGRAM     ;送字节编程命令
    MOV   IAP_TRIG,  #5Ah          ;先送5Ah,再送A5h到ISP/IAP触发寄存器,每次都需如此
    MOV   IAP_TRIG,  #0A5h         ;送完A5h后，ISP/IAP命令立即被触发起动
;CPU等待IAP动作完成后，才会继续执行程序.
    NOP                            ;字节编程成功后，CPU继续执行程序
;以下语句可不用,只是出于安全考虑而已
    MOV   IAP_CONTR, #00000000B  ;禁止ISP/IAP操作
    MOV   IAP_CMD,   #00000000B  ;去除ISP/IAP命令
    ;MOV  IAP_TRIG,  #00000000B  ;防止ISP/IAP命令误触发
    ;MOV  IAP_ADDRH, #0FFh       ;送地址高字节单元为00,指向非EEPROM区,防止误操作
    ;MOV  IAP_ADDRL, #0FFh       ;送地址低字节单元为00,指向非EEPROM区,防止误操作

;扇区擦除，没有字节擦除，只有扇区擦除，512字节/扇区,每个扇区用得越少越方便
;如果要对某个扇区进行擦除，而其中有些字节的内容需要保留，则需将其先读到单片机
;内部的RAM中保存，再将该扇区擦除，然后将须保留的数据写回该扇区，所以每个扇区
;中用的字节数越少越好，操作起来越灵活越快.
;扇区中任意一个字节的地址都是该扇区的地址,无需求出首地址.

    MOV   IAP_ADDRH, #SECTOR_FIRST_BYTE_ADDR_HIGH ;送扇区起始地址高字节 ┐ 地址需要改变时
    MOV   IAP_ADDRL, #SECTOR_FIRST_BYTE_ADDR_LOW  ;送扇区起始地址低字节 ┘ 才需重新送地址
    MOV   IAP_CONTR, #WAIT_TIME  ;设置等待时间  ┐
    ORL   IAP_CONTR, #10000000B  ;允许ISP/IAP   ┘ 此两句可以合成一句,并且只送一次就够了
    MOV   IAP_CMD,   #ISP_IAP_SECTOR_ERASE;送扇区擦除命令,命令不需改变时,不需重新送命令
    MOV   IAP_TRIG,      #5Ah      ;先送5Ah,再送A5h到ISP/IAP触发寄存器,每次都需如此
    MOV   IAP_TRIG,      #0A5h     ;送完A5h后，ISP/IAP命令立即被触发起动
;CPU等待IAP动作完成后，才会继续执行程序.
    NOP                            ;扇区擦除成功后，CPU继续执行程序
;以下语句可不用,只是出于安全考虑而已
    MOV   IAP_CONTR, #00000000B     ;禁止ISP/IAP操作
    MOV   IAP_CMD,   #00000000B     ;去除ISP/IAP命令
    ;MOV  IAP_TRIG,  #00000000B     ;防止ISP/IAP命令误触发
    ;MOV  IAP_ADDRH, #0FFh          ;送地址高字节单元为00,指向非EEPROM区
    ;MOV  IAP_ADDRL, #0FFh          ;送地址低字节单元为00,防止误操作
```

小常识：（STC 单片机的 Data Flash 当 EEPROM 功能使用）

3 个基本命令 ---- 字节读，字节编程，扇区擦除

字节编程：只能将“1”改为“0”，对“0”用字节编程是无用的。如果该字节是“1111,1111B”，则可将其中的“1 ”编程为“0 ”，如果该字节中有位为“0 ”，要将其改为“1 ”，则须先将整个扇区擦除，因为只有“扇区擦除”才可以将“0 ”变为“1 ”。

扇区擦除：只有“扇区擦除”才可能将“0 ”擦除为“1 ”。

大建议：

1.同一次修改的数据放在同一扇区中，不是同一次修改的数据放在另外的扇区,就不须读出保护。

2.如果一个扇区只用一个字节，那就是真正的 EEPROM,STC 单片机的 Data Flash 比外部 EEPROM 要快很多，读一个字节 / 编程一个字节大概是 0.2uS/60uS。

3.如果在一个扇区中存放了大量的数据，某次只需要修改其中的一个字节或一部分字节时，则另外的不需要修改的数据须先读出放在 STC 单片机的 RAM 中，然后擦除整个扇区，再将需要保留的数据和需修改的数据一并写回该扇区中。这时每个扇区使用的字节数是使用的越少越方便(不需读出一大堆需保留数据)。

5.5 一个完整的 EEPROM 测试程序，用宏晶的下载板可以直接测试

```
;STC12C5201AD 系列单片机 EEPROM/IAP 功能测试程序演示
;/* ---  STC International Limited -------------------------------------------------- */
;/* ---  宏晶科技 姚永平   设计 2006/1/6    V1.0 --------------------------------------- */
;/* ---  演示 STC12C5201AD 系列 MCU EEPROM/IAP 功能 ---------------------------------- */
;/* ---  Mobile: 13922805190 --------------------------------------------------------- */
;/* ---  Fax: 0755-82944243 ---------------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ---------------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ----------------------------------------------------- */
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过,EEPROM 的数据在 P1 口上显示
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;------------------------------------------------
;声明与 IAP/ISP/EEPROM 有关的特殊功能寄存器的地址
IAP_DATA        EQU   0C2H
IAP_ADDRH       EQU   0C3H
IAP_ADDRL       EQU   0C4H
IAP_CMD         EQU   0C5H
IAP_TRIG        EQU   0C6H
IAP_CONTR       EQU   0C7H
;定义 ISP/IAP 命令
ISP_IAP_BYTE_READ         EQU   1H  ;字节读
ISP_IAP_BYTE_PROGRAM      EQU   2H  ;字节编程,可以将 1 写成 0，要将 1 变成 0，必须执行字节编程
ISP_IAP_SECTOR_ERASE      EQU   3H  ;扇区擦除,可以将 0 擦成 1，要将 0 变成 1，必须擦除整个扇区
;定义 Flash 操作等待时间及允许 IAP/ISP/EEPROM 操作的常数
;ENABLE_IAP        EQU   80H     ;系统工作时钟<30MHz 时，对 IAP_CONTR 寄存器设置此值
;ENABLE_IAP        EQU   81H     ;系统工作时钟<24MHz 时，对 IAP_CONTR 寄存器设置此值
 ENABLE_IAP        EQU   82H     ;系统工作时钟<20MHz 时，对 IAP_CONTR 寄存器设置此值
;ENABLE_IAP        EQU   83H     ;系统工作时钟<12MHz 时，对 IAP_CONTR 寄存器设置此值
;ENABLE_IAP        EQU   84H     ;系统工作时钟<6MHz 时，对 IAP_CONTR 寄存器设置此值
;ENABLE_IAP        EQU   85H     ;系统工作时钟<3MHz 时，对 IAP_CONTR 寄存器设置此值
;ENABLE_IAP        EQU   86H     ;系统工作时钟<2MHz 时，对 IAP_CONTR 寄存器设置此值
;ENABLE_IAP        EQU   87H     ;系统工作时钟<1MHz 时，对 IAP_CONTR 寄存器设置此值
DEBUG_DATA        EQU   5AH  ;是本测试程序选定的 EEPROM 单元的数值如正确应等于的数值
;-----------------------
;选择 MCU EEPROM 测试起始地址
DATA_FLASH_START_ADDRESS EQU 0000H  ;STC12C5201AD 系列单片机的 EEPROM 测试起始地址
;------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;------------------------------------------------
    ORG   0100H
MAIN:
    MOV    P1,#0F0H          ;演示程序开始工作,让 P1.0/P1.1/P1.2/P1.3 控制的灯亮
    LCALL   Delay             ;延时
    MOV    P1,#0FH           ;演示程序开始工作,让 P1.7/P1.6/P1.5/P1.4 控制的灯亮
```

```
    LCALL    Delay             ;延时
    MOV      SP,  #7FH         ;堆栈指针指向 7FH 单元
;************************************************
;将 EEPROM 测试起始地址单元的内容读出
MAIN1:
    MOV     DPTR, #DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    LCALL   Byte_Read
    MOV      40H, A                   ;将 EEPROM 的值送 40H 单元保存
    CJNE    A, #DEBUG_DATA, DATA_NOT_EQU_DEBUG_DATA    ;如果数据比较不正确,就跳转

DATA_IS_DEBUG_DATA:
;数据是对的,亮 P1.7 控制的灯,然后在 P1 口上将 EEPROM 的数据显示出来
    MOV      P1,  #01111111B  ;如 (DATA_FLASH_START_ADDRESS)的值等于#DEBUG_DATA, 亮 P1.7
    LCALL  Delay           ;延时
    MOV      A, 40H    ;将保存在 40H 单元中 EEPROM 的值从 40H 单元送累加器 A
    CPL     A          ;取反的目的是相应的灯亮代表 1,不亮代表 0
    MOV     P1,A      ;数据是对的, 送 P1 显示
WAIT1:
    SJMP    WAIT1     ;数据是对的, 送 P1 显示后, CPU 在此无限循环执行此句

DATA_NOT_EQU_DEBUG_DATA:
;EEPROM 里的数据是错的,亮 P1.3 控制的灯,然后在 P1 口上将错误的数据显示出来,
;再将该 EEPROM 所在的扇区整个擦除,将正确的数据写入后,亮 P1.5 控制的灯
    MOV     P1,  #11110111B ;如 (DATA_FLASH_START_ADDRESS)的值不等于#DEBUG_DATA, 亮 P1.3
    LCALL   Delay        ;延时
    MOV      A,   40H  ;将保存在 40H 单元中 EEPROM 的值从 40H 单元送累加器 A
    CPL     A            ;取反的目的是相应的灯亮代表 1,不亮代表 0
    MOV     P1,  A     ;数据不对, 送 P1 显示
    LCALL    Delay;延时

    MOV     DPTR,#DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    ACALL  Sector_Erase   ;擦除整个扇区
    MOV     DPTR, #DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    MOV     A,     #DEBUG_DATA        ;写入 EEPROM 的数据为 #DEBUG_DATA
    ACALL   Byte_Program              ;字节编程
    MOV     P1,    #11011111B         ;将先前亮的 P1.3 灯关闭 ,再亮 P1.5 灯,代表数据已被修改
WAIT2:
    SJMP    WAIT2         ;字节编程后,CPU 在此无限循环执行此句
;************************************************
```

```
;-----------------------------------------------
;读一字节，调用前需打开 IAP 功能，入口:DPTR = 字节地址，返回:A = 读出字节
Byte_Read:
    MOV     IAP_CONTR,  #ENABLE_ISP      ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV     IAP_CMD,    #ISP_IAP_BYTE_READ   ;设置为 IAP/ISP/EEPROM 字节读模式命令
    MOV     IAP_ADDRH,  DPH              ;设置目标单元地址的高 8 位地址
    MOV     IAP_ADDRL,  DPL              ;设置目标单元地址的低 8 位地址
    ;CLR     EA
    MOV     IAP_TRIG,   #5AH         ;先送 5Ah,再送 A5h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV     IAP_TRIG,   #0A5H        ;送完 A5h 后，ISP/IAP 命令立即被触发起动
    NOP
    MOV     A,   IAP_DATA            ;读出的数据在 IAP_DATA 单元中,送入累加器 A
    ;SETB    EA
    ACALL IAP_Disable ;关闭 IAP 功能，清相关的特殊功能寄存器,使 CPU 处于安全状态,
                      ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
;-----------------------------------------------
;字节编程，调用前需打开 IAP 功能，入口:DPTR = 字节地址，A= 须编程字节的数据
Byte_Program:
    MOV   IAP_CONTR, #ENABLE_ISP     ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV   IAP_CMD, #ISP_IAP_BYTE_PROGRAM    ;设置为 IAP/ISP/EEPROM 字节编程模式命令
    MOV   IAP_ADDRH, DPH             ;设置目标单元地址的高 8 位地址
    MOV   IAP_ADDRL, DPL             ;设置目标单元地址的低 8 位地址
    MOV   ISP_DATA, A                ;要编程的数据先送进 ISP_DATA 寄存器
    ;CLR   EA
    MOV   IAP_TRIG, #5AH             ;先送 5Ah,再送 A5h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   IAP_TRIG, #0A5H            ;送完 A5h 后，ISP/IAP 命令立即被触发起动
    NOP
    ;SETB  EA
     ACALL IAP_Disable ;关闭 IAP 功能，清相关的特殊功能寄存器,使 CPU 处于安全状态,
                       ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
;-----------------------------------------------
;擦除扇区，入口:DPTR = 扇区地址
Sector_Erase:
    MOV   IAP_CONTR, #ENABLE_ISP     ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV   IAP_CMD, #03H              ;设置为 IAP/ISP/EEPROM 扇区擦除模式命令
    MOV   IAP_ADDRH, DPH             ;设置目标单元地址的高 8 位地址
    MOV   IAP_ADDRL, DPL             ;设置目标单元地址的低 8 位地址
    ;CLR   EA
    MOV   IAP_TRIG, #5AH             ;先送 5Ah,再送 A5h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   IAP_TRIG, #0A5H            ;送完 A5h 后，ISP/IAP 命令立即被触发起动
    NOP
    ;SETB  EA
     ACALL IAP_Disable ;关闭 IAP 功能，清相关的特殊功能寄存器,使 CPU 处于安全状态,
                       ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
```

```
;------------------------------------------------
IAP_Disable:
;关闭 IAP 功能, 清相关的特殊功能寄存器,使 CPU 处于安全状态,
;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    MOV    IAP_CONTR, #0             ;关闭 IAP 功能
    MOV    IAP_CMD,  #0              ;清命令寄存器,使命令寄存器无命令,此句可不用
    MOV    IAP_TRIG, #0              ;清命令触发寄存器,使命令触发寄存器无触发,此句可不用
    MOV    IAP_ADDRH,   #0FF             ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV    IAP_ADDRL,   #0FF             ;送地址低字节单元为 00,防止误操作
    RET
;------------------------------------------------
Delay:
    CLR    A
    MOV    R0, A
    MOV    R1, A
    MOV    R2, #20H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET

;------------------------------------------------

    END
;*************************************************
```

第六章　STC12xx 系列单片机定时器应用

6.1　定时器0/1的介绍

STC12xx 系列有 4 个定时器，其中定时器 0 和定时器 1 两个 16 位定时器，与传统 8051 的定时器完全兼容，也可以设置为 1T 模式，其中在定时器 1 做波特率发生器时，定时器 0 可以当两个 8 位定时器用(另外 2 路 PCA/PWM 可以再实现 2 个 16 位定时器)。

定时器 0 和 1

定时和计数功能由特殊功能寄存器 TMOD 的控制位 C/$\overline{T}$ 进行选择，TMOD 寄存器的各位信息如下表所列。可以看出，2 个定时 / 计数器有 4 种操作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时 / 计数器的模式 0、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

寄存器TMOD各位的功能描述



TMOD　　地址：89H　　　　复位值：00H

不可位寻址

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
定时器 1				定时器 0			

位	符号	功能
TMOD.7/	GATE	TMOD.7 控制定时器 1,置 1 时只有在 $\overline{INT1}$ 脚为高及 TR1 控制位置 1 时才可打开定时器 / 计数器 1。
TMOD.3/	GATE	TMOD.3 控制定时器 0,置 1 时只有在 $\overline{INT0}$ 脚为高及 TR0 控制位置 1 时才可打开定时器 / 计数器 0。
TMOD.6/	C/$\overline{T}$	TMOD.6 控制定时器 1 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T1/P3.5 脚输入）
TMOD.2/	C/$\overline{T}$	TMOD.2 控制定时器 0 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T0/P3.4 脚输入）

TMOD.5/TMOD.4　M1、M0 定时器定时器 / 计数器 1 模式选择

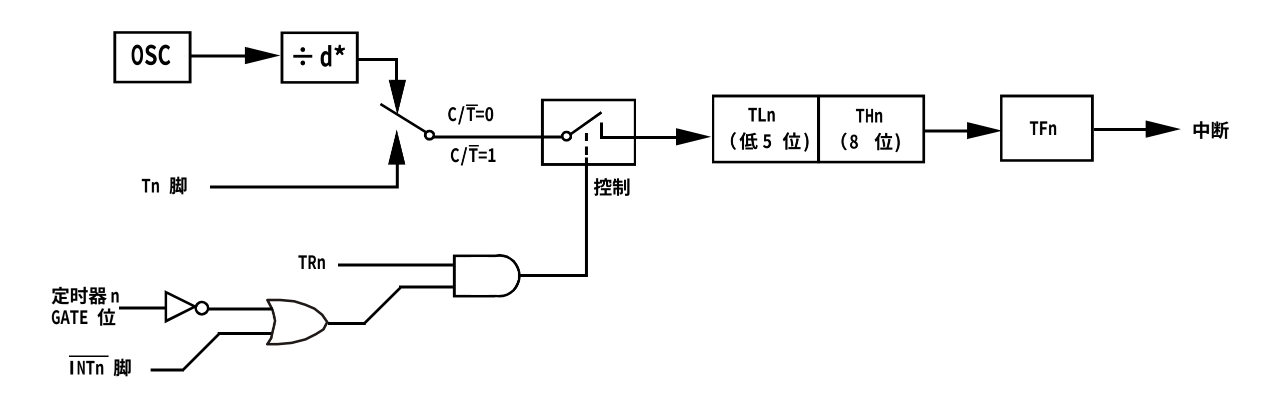
M1	M0	功能
0	0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL1 只用低 5 位参与分频，TH1 整个 8 位全用。
0	1	16 位定时器 / 计数器，TL1、TH1 全用
1	0	8 位自动重装载定时器，当溢出时将 TH1 存放的值自动重装入 TL1。
1	1	定时器 / 计数器 1 此时无效（停止计数）。

TMOD.1/TMOD.0　M1、M0 定时器 / 计数器 0 模式选择

M1	M0	功能
0	0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL0 只用低 5 位参与分频，TH0 整个 8 位全用。
0	1	16 位定时器 / 计数器，TL0、TH0 全用
1	0	8 位自动重装载定时器，当溢出时将 TH0 存放的值自动重装入 TL0。
1	1	定时器 0 此时作为双 8 位定时器 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 的控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 的控制位控制。

1.模式 0

将定时器设置成模式 0 时类似 8048 定时器，即 8 位计数器带 32 分频的预分频器。下图所示为模式 0 工作方式。此模式下，定时器配置为 13 位的计数器，由 TLn 的低 5 位和 THn 的 8 位所构成。TLn 低 5 位溢出向 THn 进位，THn 计数溢出置位 TCON 中的溢出标志位 TFn（n=0，1)。GATE=0 时，如 TRn=1，则定时器计数。GATE=1 时，允许由外部输入 $\overline{INT1}$ 控制定时器 1，$\overline{INT0}$ 控制定时器 0，这样可实现脉宽测量。TRn 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述见 TCON 寄存器各位的具体功能描述表。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 0 ： 13 位定时 / 计数器

表 寄存器 TCON 各位的功能描述

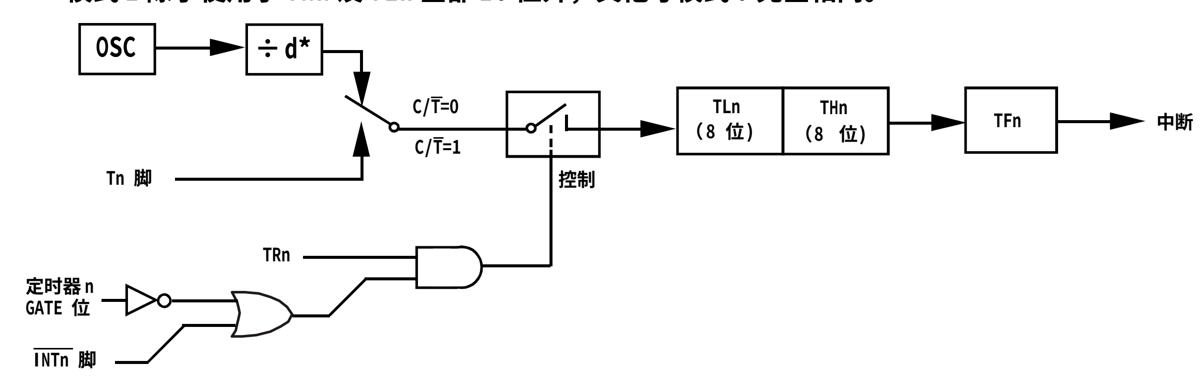
TCON 地址：88H								
可位寻址	7	6	5	4	3	2	1	0
复位值：00H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

位	符 号	功 能
TCON.7	TF1	定时器 / 计数器 1 溢出标志位。当 T1 被允许计数后，T1 从初值开始加 1 计数，最高位产生溢出时，置“1”TF1，并向 CPU 请求中断，当 CPU 响应时，由硬件清“0”TF1，TF1 也可以由程序查询或清“0”。
TCON.6	TR1	定时器 T1 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.7）=0，TR1=1 时就允许 T1 开始计数，TR1=0 时禁止 T1 计数。当 GATE（TMOD.7）=1，TR1=1 且 INT1 输入高电平时，才允许 T1 计数。
TCON.5	TF0	定时器 / 计数器 0 溢出标志位。当 T0 被允许计数后，T0 从初值开始加 1 计数，最高位产生溢出时，置“1”TF0，并向 CPU 请求中断，当 CPU 响应时，由硬件清“0”TF0，TF0 也可以由程序查询或清“0”。
TCON.4	TR0	定时器 T0 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.3）=0，TR0=1 时就允许 T0 开始计数，TR1=0 时禁止 T0 计数。当 GATE（TMOD.3）=1，TR0=1 且 $\overline{\text{INT0}}$ 输入高电平时，才允许 T0 计数。
TCON.3	IE1	外部中断 1 中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将 IE1 位清 0。
TCON.2	IT1	外部中断 1 触发方式控制位。IT1=0 时，外部中断 1 为低电平触发方式，当 $\overline{\text{INT1}}$（P3.3）输入低电平时，置位 IE1。采用低电平触发方式时，外部中断源（输入到 INT1）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.3 要变高），否则将产生另一次中断。当 IT1=1 时，则外部中断 1（$\overline{\text{INT1}}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位 IE1，向主机请求中断处理。
TCON.1	IE0	外部中断 0 中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将 IE0 位清 0。
TCON.0	IT0	外部中断 0 触发方式控制位。IT0=0 时，外部中断 0 为低电平触发方式，当 $\overline{\text{INT0}}$（P3.2）输入低电平时，置位 IE0。采用低电平触发方式时，外部中断源（输入到 $\overline{\text{INT0}}$）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.2 要变高），否则将产生另一次中断。当 IT0=1 时，则外部中断 0（$\overline{\text{INT0}}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位 IE1，向主机请求中断处理。

该 13 位寄存器包含 THn 全部 8 个位及 TLn 的低 5 位。TLn 的高 3 位不定，可将其忽略。置位运行标志（TRn）不能清零此寄存器。模式 0 的操作对于定时器 0 及定时器 1 都是相同的。2 个不同的 GATE 位（TMOD.7 和 TMOD.3）分别分配给定时器 1 及定时器 0。

2.模式 1

模式 1 除了使用了 THn 及 TLn 全部 16 位外，其他与模式 0 完全相同。

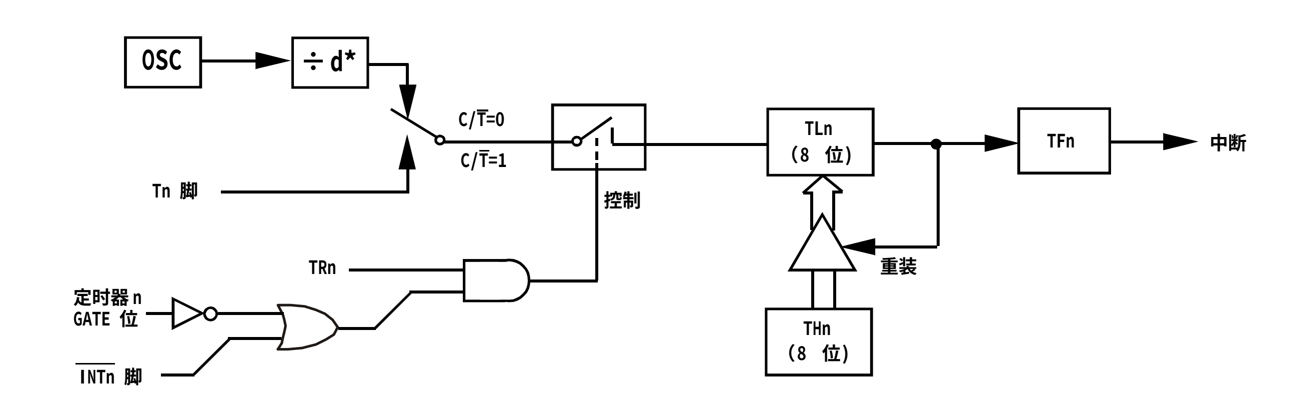


* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 1 ： 16 位定时 / 计数器

3.模式 2

此模式下定时器 / 计数器 0 和 1 作为可自动重装载的 8 位计数器（TLn），如下图所示。TLn 的溢出不仅置位 TFn，而且将 THn 内容重新装入 TLn，THn 内容由软件预置，重装时 THn 内容不变。模式 2 的操作对于定时器 0 及定时器 1 是相同的。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

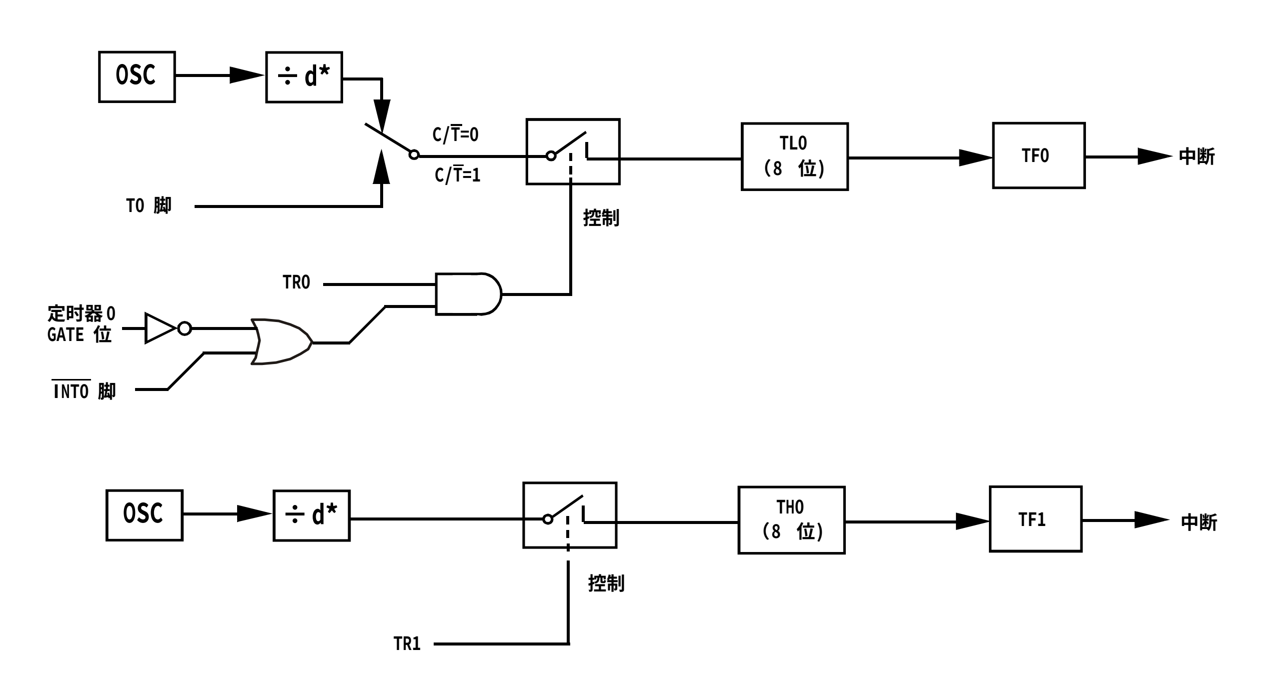
图 定时器 / 计数器 0 和 1 的模式 2 ： 8 位自动重装载

4.模式 3

对定时器 1，在模式 3 时，定时器 1 停止计数，效果与将 TR1 设置为 0 相同。

对定时器 0，此模式下定时器 0 的 TL0 及 TH0 作为 2 个独立的 8 位计数器。下图为模式 3 时的定时器 0 逻辑图。TL0 占用定时器 0 的控制位：C/T̄、GATE、TR0、$\overline{INT0}$ 及 TF0。TH0 限定为定时器功能（计数器周期），占用定时器 1 的 TR1 及 TF1。此时，TH0 控制定时器 1 中断。

模式 3 是为了增加一个附加的 8 位定时器 / 计数器而提供的，使单片机具有三个定时器 / 计数器。模式 3 只适用于定时器 / 计数器 0，定时器 T1 处于模式 3 时相当于 TR1=0，停止计数（此时 T1 可用来作串行口波特率发生器），而 T0 可作为两个定时器用。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)；　在 T0x12 = 1 模式下，d=1(1T)。

图　定时 / 计数器 0 的模式 3　：　两个 8 位计数器

5.也可将定时器 0 和定时器 1 设置为 1T 模式

STC12C5201AD 系列是 1T 的 8051 单片机，为兼容传统 8051，定时器 0 和定时器 1 复位后是传统 8051 的速度，即 12 分频，这是为了兼容传统 8051。但也可不进行 12 分频，通过设置新增加的特殊功能寄存器 AUXR，将 T0，T1 设置为 1T。普通 111 条机器指令是固定的，快 3 到 24 倍，无法改变。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	-	-	-	-	-	0000,000

T0x12:　0，定时器 0 是传统 8051 速度，12 分频；1，定时器 0 的速度是传统 8051 的 12 倍，不分频
T1x12:　0，定时器 1 是传统 8051 速度，12 分频；1，定时器 1 的速度是传统 8051 的 12 倍，不分频
如果 UART 串口用定时器 1 做波特率发生器，T1x12 位就可以控制 UART 串口是 12T 还是 1T 了。

UART 串口的模式 0:
STC12C5201AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，UART 串口复位后是兼容传统 8051 的。
UART_M0x6:　0，UART 串口的模式 0 是传统 12T 的 8051 速度，12 分频；
　　1，UART 串口的模式 0 的速度是传统 12T 的 8051 的 6 倍，2 分频
如果用定时器 T1 做波特率发生器时，UART 串口的速度由 T1 的溢出率决定

6.2 定时器0/1应用举例

【例1】 定时 / 计数器编程，定时 / 计数器的应用编程主要需考虑：根据应用要求，通过程序初始化，正确设置控制字，正确计算和计算计数初值，编写中断服务程序，适时设置控制位等。通常情况下，设置顺序大致如下：

1)工作方式控制字（TMOD、T2CON）的设置；

2)计数初值的计算并装入THx、TLx、RCAP2H、RCAP2L；

3)中断允许位ETx、EA的设置，使主机开放中断；

4) 启 / 停位TRx的设置等。

现以定时 / 计数器0或1为例作一简要介绍。

8051系列单片机的定时器 / 计数器0或1是以不断加1进行计数的，即属加1计数器，因此，就不能直接将实际的计数值作为计数初值送入计数寄存器THx、TLx中去，而必须将实际计数值以2^8、2^{13}、2^{16}为模求补，以其补码作为计数初值设置THx和TLx。

设：实际计数值为X，计数器长度为n（n=8、13、16），则应装入计数器THx、TLx中的计数初值为2^n-x，式中2^n为取模值。例如，工作方式0的计数长度为13位，则n=13，以2^{13}为模，工作方式1的计数长度为16，则n=16，以2^{16}为模等等。所以，计数初值为（x）= 2^n-x。

对于定时模式，是对机器周期计数，而机器周期与选定的主频密切相关。因此，需根据应用系统所选定的主频计算出机器周期值。现以主频6MHz为例，则机器周期为：

$$\text{一个机器周期} = \frac{12}{\text{主振频率}} = \frac{12}{6\times 10^6}\ \mu s = 2\ \mu s$$

实际定时时间Tc = x·Tp

式中Tp为机器周期，Tc为所需定时时间，x为所需计数次数。Tp和Tp一般为已知值，在求出Tp后即可求得所需计数值x，再将x求补码，即求得定时计数初值。即

(x）补 = 2^n - x

例如，设定时时间Tc = 5ms，机器周期TP = 2μs，可求得定时计数次数

$$x = \frac{5ms}{2\ \mu s} = 2500\text{次}$$

设选用工作方式1，则n=16，则应设置的定时时间计数初值为：(x）补 = 2^{16} - x=65536-2500=63036，还需将它分解成两个8位十六进制数，分别求得低8位为3CH装入TLx，高8位为F6H装入THx中。

工作方式0、1、2的最大计数次数分别为8192、65536和256。

对外部事件计数模式，只需根据实际计数次数求补后变换成两个十六进制码即可。

【例2】 定时 / 计数器应用编程，设某应用系统，选择定时 / 计数器1定时模式，定时时间Tc = 10ms，主频频率为12MHz，每10ms向主机请求处理。选定工作方式1。计算得计数初值：低8位初值为F0H，高8位初值为D8H。

(1) 初始化程序

所谓初始化，一般在主程序中根据应用要求对定时 / 计数器进行功能选择及参数设定等预置程序，本例初始化程序如下：

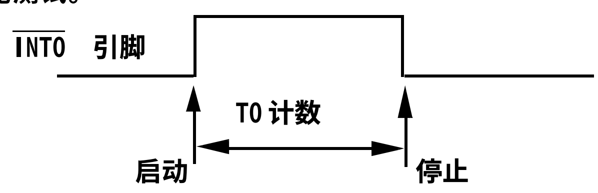
```
START：
        ⋮                          ；主程序段
        MOV   SP，#60H             ；设置堆栈区域
        MOV   TMOD，#10H           ；选择 T1、定时模式，工作方式 1
        MOV   TH1，#0D8H           ；设置高字节计数初值
        MOV   TL1，#0F0H           ；设置低字节计数初值
        SETB  EA                   } 开中断
        SETB  ET1                  }
        ⋮                          ；其他初始化程序
        SETB  TR1                  ；启动 T1 开始计时
        ⋮                          ；继续主程序
```

（2）中断服务程序

```
INTT1：  PUSH  A                   }
         PUSH DPL                  } 现场保护
         PUSH DPH                  }
          ⋮
         MOV   TL1,#0F0H           } 重新置初值
         MOV   TH1,#0D8H           }
          ⋮                        ；中断处理主体程序
         POP   DPH                 }
         POP   DPL                 } 现场恢复
         POP   A                   }
         RETI                      ；返回
```

这里展示了中断服务子程序的基本格式。8052 系列单片机的中断属于矢量中断，每一个矢量中断源只留有 8 个字节单元，一般是不够用的，常需用转移指令转到真正的中断服务子程序区去执行。

【例 3】 对外部正脉冲测宽。选择定时 / 计数器 2 进行脉宽测试较方便，但也可选用定时 / 计数器 0 或定时 / 计数器 1 进行测宽操作。本例选用定时 / 计数器 0（T0）以定时模式，工作方式 1 对 $\overline{\text{INT0}}$ 引脚上的正脉冲进行脉宽测试。



设置 GATE 为 1，机器周期 TP 为 1 μs。本例程序段编制如下：

```
INTT0：   MOV   TMOD，#09H        ；设 T0 为定时方式 1，GATE 为 1
```

```
            MOV   TL0, #00H            ;}
            MOV   TH0, #00H            ;} TH0,TL0清0
            CLR   EX0                  ;关INT0中断
LOP1:       JB    P3.2, LOP1       ;等待INT0引低电平
LOP2:       JNB   P3.2, LOP2       ;等待INT0引脚高电平
            SETB  TR0                  ;启动T0开始计数
LOP3:       JB    P3.2, LOP3       ;等待INT0低电平
            CLR   TR0                  ;停止T0计数
            MOV   A, TL0               ;低字节计数值送A
            MOV   B, TH0               ;高字节计数值送B
             ⋮                         ;计算脉宽和处理
```

【例4】 利用定时/计数器0或定时/计数器1的Tx端口改造成外部中断源输入端口的应用设计。

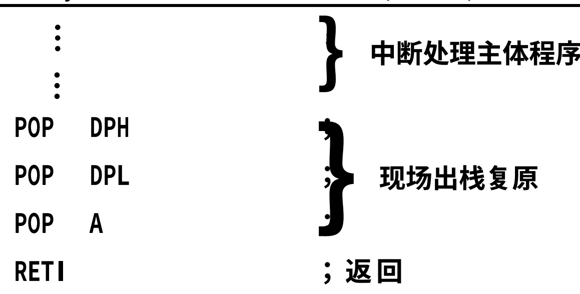
在某些应用系统中常会出现原有的两个外部中断源INT0和INT1不够用，而定时/计数器有多余，则可将Tx用于增加的外部中断源。现选择定时/计数器1为对外部事件计数模式工作方式2（自动再装入），设置计数初值为FFH，则T1端口输入一个负跳变脉冲，计数器即回0溢出，置位对应的中断请求标志位TF1为1，向主机请求中断处理，从而达到了增加一个外部中断源的目的。应用定时/计数器1（T1）的中断矢量转入中断服务程序处理。其程序示例如下：

（1）主程序段：

```
      ORG   0000H
      AJMP  MAIN              ;转主程序
      ORG   001BH
      LJMP  INTER             ;转T1中断服务程序
       ⋮
      ORG   0100              ;主程序入口
MAIN: …
       ⋮
      MOV   SP, #60H          ;设置堆栈区
      MOV   TMOD, #60H;设置定时/计数器1，计数方式2
      MOV   TL1, #0FFH;设置计数常数
      MOV   TH1, #0FFH
      SETB  EA                ;开中断
      SETB  ET1               ;开定时/计数器1中断
      SETB  TR1               ;启动定时/计数器1计数
       ⋮
```

（2）中断服务程序（具体处理程序略）

```
            ORG   1000H
INTER:      PUSH  A            ;}
            PUSH  DPL          ;} 现场入栈保护
            PUSH  DPH          ;}
             ⋮
```



这是中断服务程序的基本格式。

【例 5】 某应用系统需通过 P1.0 和 P1.1 分别输出周期为 200 μs 和 400 μs 的方波。为此，系统选用定时器 / 计数器 0（T0），定时方式 3，主频为 6MHz，TP=2 μs，经计算得定时常数为 9CH 和 38H。

本例程序段编制如下：

（1）初始化程序段

```
           ⋮
PLT0:MOV   TMOD,#03H        ;设置 T0 定时方式 3
     MOV   TL0，#9CH        ;设置 TL0 初值
     MOV   TH0，#38H        ;设置 TH0 初值
     SETB  EA               ;
     SETB  ET0              ;  } 开中断
     SETB  ET1              ;
     SETB  TR0              ;启动
     SETB  TR1              ;启动
           ⋮
```

(2)中断服务程序段

1)

```
INTOP：  ⋮
         ⋮
     MOV   TL0，#9CH        ;重新设置初值
     CPL   P1.0             ;对 P1.0 输出信号取反
         ⋮
     RETI                   ;返回
```

2)

```
INT1P    ⋮
         ⋮
     MOV   TH0，#38H        ;重新设置初值
     CPL   P1.1             ;对 P1.1 输出信号取反
         ⋮
     RETI                   ;返回
```

在实际应用中应注意的问题如下。

（1）定时 / 计数器的实时性

定时 / 计数器启动计数后，当计满回 0 溢出向主机请求中断处理，由内部硬件自动进行。但从回 0 溢出请求中断到主机响应中断并作出处理存在时间延迟，且这种延时随中断请求时的现场环境的不同而不同，一般需延时 3 个机器周期以上，这就给实时处理带来误差。大多数应用场合可忽略不计，但对某些要求实时性苛刻的场合，应采用补偿措施。

这种由中断响应引起的时间延时，对定时 / 计数器工作于方式 0 或 1 而言有两种含义：一是由于中断响应延时而引起的实时处理的误差；二是如需多次且连续不间断地定时 / 计数，由于中断响应延时，则在中断服务程序中再置计数初值时已延误了若干个计数值而引起误差，特别是用于定时就更明显。

例如选用定时方式 1 设置系统时钟，由于上述原因就会产生实时误差。这种场合应采用动态补偿办法以减少系统始终误差。所谓动态补偿，即在中断服务程序中对 THx、TLx 重新置计数初值时，应将 THx、TLx 从回 0 溢出又重新从 0 开始继续计数的值读出，并补偿到原计数初值中去进行重新设置。可考虑如下补偿方法：

```
    ⋮
CLR   EA           ；禁止中断
MOV   A，TLx       ；读 TLx 中已计数值
ADD   A，#LOW      ；LOW 为原低字节计数初值
MOV   TLx，A       ；设置低字节计数初值
MOV   A，#HIGH     ；原高字节计数初值送 A
ADDC  A，THx       ；高字节计数初值补偿
MOV   THx，A       ；置高字节计数初值
SETB  EA           ；开中断
    ⋮
```

（2）动态读取运行中的计数值

在动态读取运行中的定时 / 计数器的计数值时，如果不加注意，就可能出错。这是因为不可能在同一时刻同时读取 THx 和 TLx 中的计数值。比如，先读 TLx 后读 THx，因为定时 / 计数器处于运行状态，在读 TLx 时尚未产生向 THx 进位，而在读 THx 前已产生进位，这时读得的 THx 就不对了；同样，先读 THx 后读 TLx 也可能出错。

一种可避免读错的方法是：先读 THx，后读 TLx，将两次读得的 THx 进行比较；若两次读得的值相等，则可确定读的值是正确的，否则重复上述过程，重复读得的值一般不会再错。此法的软件编程如下：

```
RDTM：MOV  A，THx             ；读取 THx 存 A 中
      MOV  R0，TLx            ；读取 TLx 存 R0 中
      CJNE A，THx，RDTM       ；比较两次 THx 值，若相等，则读得的值正
                              ；确，程序往下执行，否则重读
      MOV  R1，A              ；将 THx 存于 R1 中
       ⋮
```

6.3 用定时器1做波特率发生器

```
;/* ---  STC International Limited ---------------------------------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 --------------------------------------- */
;/* ---  演示 STC12xx 系列 MCU 定时器1作波特率发生器功能 ------------------------------ */
;/* ---  Mobile: 13922805190 ---------------------------------------------------------- */
;/* ---  Fax: 0755-82944243 ----------------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ----------------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ------------------------------------------------------ */
;本演示程序在宏晶的STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;---------------------------------------------------------------------------------
;     本程序演示STC12xx 系列单片机用定时器1作 RS-232 通信
;波特率发生器的使用方法，有关波特率自动重装数的计算请查看程序后面的内容
;     STC12xx 系列是 "一个时钟/机器周期" 的 8051单片机。它
;的定时器0、定时器1 有两种计数速率，一种是12T模式：每12个时钟加 1，与普通的
;8051单片机相同；另一种是1T模式：每个时钟加 1，是普通8051单片机的 12 倍。
;     STC89C51RC/RD+ 系列是 "12个时钟/机器周期" 的 8051单片机，与普通的 8051单片
; 机相同。
;     STC12xx 系列的单片机，定时器0、定时器1 的计数速率由
;特殊功能寄存器 AUXR 的 bit7，bit6 决定，bit7 的符号是 T0x12，如果 T0x12=1，
;定时器0 就工作在 1T模式。bit6 的符号是 T1x12，如果 T1x12=1，定时器1 就工作在
;1T模式。有关详情请参考 STC12C5201AD 系列单片机器件手册(中文应用指南)。

;使用方法:
; 1. 修改程序，改变波特率参数或改变定时器1的计数速率(1T模式/12T模式)
; 2. 汇编程序，将代码下载到单片机中
; 3. 调整串口调试助手的波特率与单片机的波特率相同，并打开调试助手的串口。STC
;     下载程序 STC-ISP.exe 版本 3.2 以上有串口调试助手功能。
; 4. 打开单片机电源，可以在串口调试助手的接收区看到单片机发出的数据
; 5. 用串口调试助手发送单个字节到单片机，单片机收到后会立即回发到串口调试助手
; 6. 反复步骤 1-5，检验波特率参数是否正确，特别要观察定时器1 工作在 1T模式
;     的波特率。例如，先设置定时器1 工作在 12T模式，设置波特率为 9600，执行
;     步骤 2-5，检验波特率参数是否正确。然后仅仅将定时器1 的计数速率改成
;     1T模式，执行步骤 2-5，就会发现本程序的波特率变成了 115200，波特率是
;     12T模式的 12 倍。
;
;---------------------------------------------------------------------------------
```

```
;定义 STC12xx 系列 MCU 特殊功能寄存器
AUXR    EQU  8EH
;-----------------------------------------------------------------------------
;定义波特率自动重装数
;*****************************************************************************
;以下是 Fosc = 22.1184MHz, 1T 模式, SMOD=1 时，计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=1,382,400 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=691,200 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=460,800 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=345,600 bps
;RELOAD_COUNT EQU 0FBH      ;Baud=276,480 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=230,400 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0DCH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0B8H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 70H      ;Baud=9,600 bps
;以上是 Fosc = 22.1184MHz, 1T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*****************************************************************************

;*****************************************************************************
;以下是 Fosc = 1.8432MHz, 1T 模式, SMOD=1 时，计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0D0H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0A0H      ;Baud=1,200 bps
;以上是 Fosc = 1.8432MHz, 1T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*****************************************************************************

;*****************************************************************************
;以下是 Fosc = 18.432MHz, 1T 模式, SMOD=1 时，计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=1,152,000 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=576,000 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=288,000 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=144,000 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0E2H      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0C4H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 088H      ;Baud=9,600 bps
;以上是 Fosc = 18.432MHz, 1T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*****************************************************************************
```

```
;*******************************************************************************
;以下是 Fosc = 18.432MHz, 1T 模式, SMOD=0 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=576,000 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=288,000 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=144,000 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0E2H      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0C4H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 088H      ;Baud=4,800 bps

;以上是 Fosc = 18.432MHz, 1T 模式, SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 18.432MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率

RELOAD_COUNT EQU 0FBH      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=1,200 bps

;以上是 Fosc = 18.432MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 18.432MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FBH      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0B0H      ;Baud=1,200 bps

;以上是 Fosc = 18.432MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*******************************************************************************
```

```
;*******************************************************************************
;以下是 Fosc = 11.0592MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=14,400 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=1,200 bps

;以上是 Fosc = 11.0592MHz, 12T 模式, SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 11.0592MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=14,400 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0D0H      ;Baud=1,200 bps

;以上是 Fosc = 11.0592MHz, 12T 模式, SMOD=1 时，计算出的自动重装数和波特率
;*******************************************************************************

;定义指示灯
LED_MCU_START   EQU   P1.7            ;MCU 工作指示灯
;-------------------------------------------------------------------------------
     ORG    0000H
    AJMP   MAIN
;-------------------------------------------------------------------------------
    ORG    0023H
    AJMP   UART_Interrupt             ;RS232 串口中断服务程序
    NOP
    NOP
;-------------------------------------------------------------------------------
MAIN:
    MOV    SP, #7FH                   ;设置堆栈指针
    CLR    LED_MCU_START              ;点亮 MCU 工作指示灯
    ACALL  Initial_UART               ;初始化串口
    MOV    R0, #30H                   ;30H = 可打印字符 '0' 的 ASCII 码
    MOV    R2, #10                    ;发送 10 个字符 '0123456789'
```

```
LOOP:
    MOV    A, R0
    ACALL  Send_One_Byte              ;发送一个字节,可将 PC 串口调试助手设置成字符显示
    ;如果是字符显示, 显示为 0123456789,
    ;如设置成 16 进制显示, 显示 30 31 32 33 34 35 36 37 38 39
    INC    R0
    DJNZ   R2, LOOP
MAIN_WAIT:
    SJMP   MAIN_WAIT                  ;跳转到本行, 无限循环
;-------------------------------------------------------------------------------
UART_Interrupt:                       ;串口中断服务程序
    JB     RI, Is_UART_Receive
    CLR    TI                         ;清零串口发送中断标志
    RETI                               ;发送时使用的是查询方式, 不使用中断
Is_UART_Receive:
    CLR    RI
    PUSH   ACC
    MOV    A, SBUF                    ;取接收到的字节
    ACALL  Send_One_Byte              ;回发收到的字节
    POP    ACC
    RETI
;-------------------------------------------------------------------------------
Initial_UART:                         ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1    0
;            SM0/FE   SM1   SM2   REN   TB8   RB8   TI   RI
    MOV    SCON, #50H                 ; 0101,0000 8 位可变波特率, 无奇偶校验

    MOV    TMOD, #21H                 ;设置定时器 1 为 8 位自动重装计数器
    MOV    TH1, #RELOAD_COUNT         ;设置定时器 1 自动重装数
    MOV    TL1, #RELOAD_COUNT

;---------------------------------------
;    ORL    PCON, #80H                ;若本行有效, 波特率可以加倍
;---------------------------------------
;以下两行指令只能有一行有效
;    ORL    AUXR, #01000000B           ;定时器 1 工作在 1T 模式, 波特率可以快 12 倍
     ANL    AUXR, #10111111B           ;定时器 1 工作在 12T 模式, 与普通的 8051 相同
;以上两行指令只能有一行有效
;---------------------------------------
    SETB   TR1                        ;启动定时器 1
    SETB   ES
    SETB   EA
    RET
```

```
;------------------------------------------------------------------------------
;入口参数: A = 要发送的字节
Send_One_Byte:                          ;发送一个字节
    CLR   ES
    CLR   TI                            ;清零串口发送中断标志
    MOV   SBUF, A
Wait_Send_Finish:
    JNB   TI, Wait_Send_Finish          ;等待发送完毕
    CLR   TI                            ;清零串口发送中断标志
    SETB  ES
    RET
;------------------------------------------------------------------------------
    END
;------------------------------------------------------------------------------
;计算自动重装数 RELOAD (SMOD = 0, SMOD 是 PCON 特殊功能寄存器的最高位):
;  1. 计算 RELOAD (以下是 SMOD = 0 时的计算公式)
;
;     a) 12T 模式的计算公式: RELOAD = 256 - INT(Fosc/Baud0/32/12 + 0.5)
;     b)  1T 模式的计算公式: RELOAD = 256 - INT(Fosc/Baud0/32 + 0.5)
;
;     式中: INT() 表示取整运算即舍去小数, 在式中加 0.5 可以达到四舍五入的目的
;           Fosc = 晶振频率
;           Baud0 = 标准波特率
;
;  2. 计算用 RELOAD 产生的波特率:
;       a)   Baud = Fosc/(256 - RELOAD)/32/12        12T 模式
;       b)   Baud = Fosc/(256 - RELOAD)/32            1T 模式
;
;  3. 计算误差
;          error = (Baud - Baud0)/Baud0 * 100%
;  4. 如果误差绝对值 > 3% 要更换波特率或者更换晶体频率, 重复步骤 1-4
;
;
;例: Fosc = 22.1184MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 22118400/57600/32/12 + 0.5)
;           = 256 - INT( 1.5 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 22118400/(256-255)/32/12
;         = 57600
; 3. 误差等于零
```

```
;例: Fosc = 18.432MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/57600/32/12 + 0.5)
;           = 256 - INT( 0.833 + 0.5 )
;           = 256 - INT( 1.333 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 18432000/(256-255)/32/12
;         = 48000
; 3. error = (48000 - 57600)/57600 * 100%
;          = -16.66%
; 4. 误差很大，要更换波特率或者更换晶体频率，重新计算请见下一例

;例: Fosc = 18.432MHz, Baud0 = 9600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/9600/32/12 + 0.5)
;           = 256 - INT( 5.5 )
;           = 256 - 5
;           = 251
;           = 0FBH
; 2. Baud = 18432000/(256-251)/32/12
;         = 9600
; 3. 一目了然，误差等于零

;例: Fosc = 2.000MHz, Baud = 4800 (1T 模式)
; 1. RELOAD = 256 - INT( 2000000/4800/32 + 0.5)
;           = 256 - INT( 13.02 + 0.5 )
;           = 256 - INT( 13.52 )
;           = 256 - 13
;           = 243
;           = 0F3H
; 2. Baud = 2000000/(256-243)/32
;         = 4808
; 3. error = 0.16%
;-----------------------------------------------------------------------------
```

第七章 STC12 系列单片机的 A/D 转换

7.1 STC12C5201AD系列单片机A/D转换相关寄存器

STC12C5201AD 系列带 A/D 转换的单片机的 A/D 转换口在 P1 口(P1.7-P1.0)，有 8 路 8 位高速 A/D 转换器，速度可达到 300KHz(30 万次 / 秒)。8 路电压输入型 A/D，可做温度检测、电池电压检测、按键扫描、频谱检测等。上电复位后 P1 口为弱上拉型 I/O 口，用户可以通过软件设置将 8 路中的任何一路设置为 A/D 转换，不需作为 A/D 使用的口可继续作为 I/O 口使用。

需作为 A/D 使用的口需先将 P1ASF 特殊功能寄存器中的相应位置为 ‘ 1 ’，将相应的口设置为模拟功能。

STC12C5202AD 系列单片机 P1 口模拟功能控制寄存器(该寄存器是只写寄存器,读无效)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000

当 P1 口中的相应位作为 A/D 使用时,要将 P1ASF 中的相应位置 1.

P1ASF[7:0]	P1.x的功能	其中P1ASF寄存器地址为：[9DH] （不能够进行位寻址）
P1ASF.0 = 1	P1.0口作为模拟功能A/D使用	
P1ASF.1 = 1	P1.1口作为模拟功能A/D使用	
P1ASF.2 = 1	P1.2口作为模拟功能A/D使用	或P1.2口作为比较器用时，在Power_Down模式下低功耗
P1ASF.3 = 1	P1.3口作为模拟功能A/D使用	
P1ASF.4 = 1	P1.4口作为模拟功能A/D使用	
P1ASF.5 = 1	P1.5口作为模拟功能A/D使用	
P1ASF.6 = 1	P1.6口作为模拟功能A/D使用	
P1ASF.7 = 1	P1.7口作为模拟功能A/D使用	

与 A/D 转换有关的特殊功能控制寄存器表

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000
ADC_CONTR	BCh	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000,0000
ADC_RES	BDh	A/D 转换结果寄存器									0000,0000
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

如果要允许 A/D 转换中断则需要将相应的控制位置 1:

1、将 EADC 置 1，允许 ADC 中断，这是 ADC 中断的中断控制位。

2、将 EA 置 1，打开单片机总中断控制位，此位不打开，也是无法产生 ADC 中断的

A/D 中断服务程序中要用软件清 A/D 中断请求标志位 ADC_FLAG(也是 A/D 转换结束标志位)。

ADC_CONTR 特殊功能寄存器： A/D 转换控制特殊功能寄存器,地址在 0BCh 单元

A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000,0000

对 ADC_CONTR 寄存器进行操作，建议直接用 MOV 赋值语句，不要用 ‘ 与 ’ 和 ‘ 或 ’ 语句

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Channel Select 模拟输入通道选择
0	0	0	选择 P1.0 作为 A/D 输入来用
0	0	1	选择 P1.1 作为 A/D 输入来用
0	1	0	选择 P1.2 作为 A/D 输入来用
0	1	1	选择 P1.3 作为 A/D 输入来用
1	0	0	选择 P1.4 作为 A/D 输入来用
1	0	1	选择 P1.5 作为 A/D 输入来用
1	1	0	选择 P1.6 作为 A/D 输入来用
1	1	1	选择 P1.7 作为 A/D 输入来用

ADC_START: 模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换,转换结束后为0。

ADC_FLAG: 模数转换器转换结束标志位,当A/D转换完成后，ADC_FLAG = 1，要由软件清0。

不管是A/D转换完成后由该位申请产生中断，还是由软件查询该标志位A/D转换是否结束，当A/D转换完成后，ADC_FLAG = 1，一定要软件清0。

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D转换所需时间
1	1	70个时钟周期转换一次，CPU工作频率21MHz时，A/D转换速度约300KHz
1	0	140个时钟周期转换一次
0	1	280个时钟周期转换一次
0	0	420个时钟周期转换一次

STC12C5202AD系列单片机的A/D转换模块说使用的时钟是外部晶体时钟或内部R/C振荡器所产生的系统时钟，不使用时钟分频寄存器CLK_DIV对系统时钟分频后所产生的供给CPU工作所使用的时钟.

好处:

这样可以让ADC用较高的频率工作,提高A/D的转换速度

这样可以让CPU用较低的频率工作,降低系统的功耗

程序中需要注意的事项:

由于是2套时钟,所以,设置ADC_CONTR控制寄存器后,要加4个空操作延时才可以正确读到ADC_CONTR寄存器的值,原因是设置ADC_CONTR控制寄存器的语句执行后,要经过4个CPU时钟的延时,其值才能够保证被设置进ADC_CONTR控制寄存器.

```
MOV  ADC_CONTR, #DATA
NOP
NOP
NOP
NOP
MOV  A,   ADC_CONTR     ;经过4个时钟延时后,才能够正确读到ADC_CONTR控制寄存器的值
```

ADC_POWER: ADC电源控制位。

0：关闭ADC电源；1：打开A/D转换器电源.建议进入空闲模式前，将ADC电源关闭，ADC_POWER =0.
启动AD转换前一定要确认AD电源已打开，AD转换结束后关闭AD电源可降低功耗，也可不关闭。
初次打开内部A/D转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动A/D转换
建议启动A/D转换后，在A/D转换结束之前，不改变任何I/O口的状态，有利于高精度A/D转换

ADC_RES特殊功能寄存器: A/D转换结果特殊功能寄存器

ADC_RES	BDh	A/D转换结果寄存器									0000,0000

模拟/数字转换结果计算公式如下：结果 ADC_RES[7:0] = 256 x Vin / Vcc

Vin为模拟输入通道输入电压，Vcc为单片机实际工作电压，用单片机工作电压作为模拟参考电压。

7.2 STC12C5A60AD/S2系列单片机A/D转换相关寄存器

STC12C5A60AD/S2系列带A/D转换的单片机的A/D转换口在P1口(P1.7-P1.0)，有8路10位高速A/D转换器,速度可达到250KHz(25万次/秒)。8路电压输入型A/D，可做温度检测、电池电压检测、按键扫描、频谱检测等。上电复位后P1口为弱上拉型I/O口，用户可以通过软件设置将8路中的任何一路设置为A/D转换，不需作为A/D使用的口可继续作为I/O口使用。

需作为A/D使用的口需先将P1ASF特殊功能寄存器中的相应位置为'1'，将相应的口设置为模拟功能。

STC12C5A60AD/S2系列单片机P1口模拟功能控制寄存器(该寄存器是只写寄存器,读无效)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000

当P1口中的相应位作为A/D使用时,要将P1ASF中的相应位置1.

P1ASF[7:0]	P1.x的功能	其中P1ASF寄存器地址为：[9DH] （不能够进行位寻址）
P1ASF.0 = 1	P1.0口作为模拟功能A/D使用	
P1ASF.1 = 1	P1.1口作为模拟功能A/D使用	
P1ASF.2 = 1	P1.2口作为模拟功能A/D使用	
P1ASF.3 = 1	P1.3口作为模拟功能A/D使用	
P1ASF.4 = 1	P1.4口作为模拟功能A/D使用	
P1ASF.5 = 1	P1.5口作为模拟功能A/D使用	
P1ASF.6 = 1	P1.6口作为模拟功能A/D使用	
P1ASF.7 = 1	P1.7口作为模拟功能A/D使用	

与A/D转换有关的特殊功能控制寄存器表

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1ASF	9Dh	P1 Analog Special Function	P17ASF	P16ASF	P15ASF	P14ASF	P13ASF	P12ASF	P11ASF	P10ASF	0000,0000
ADC_CONTR	BCh	A/D转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000,0000
ADC_RES	BDh	A/D转换结果寄存器									0000,0000
ADC_RESL	BEh	A/D转换结果寄存器低									0000,0000
AUXR1	A2h	Auxiliary register 1		PCA_P4	SPI_P4	S2_P4	GF2	ADRJ		DPS	0000,0000
IE	A8h	Interrupt Enable	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0	0000,0000
IPH	B7h	Interrupt Priority High	PPCAH	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

如果要允许A/D转换中断则需要将相应的控制位置1:

1、将EADC置1，允许ADC中断，这是ADC中断的中断控制位。

2、将EA置1，打开单片机总中断控制位，此位不打开，也是无法产生ADC中断的

A/D中断服务程序中要用软件清A/D中断请求标志位ADC_FLAG(也是A/D转换结束标志位)。

ADC_CONTR 特殊功能寄存器： A/D转换控制特殊功能寄存器,地址在0BCh单元

A/D转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0000,0000

对ADC_CONTR寄存器进行操作，建议直接用MOV赋值语句，不要用'与'和'或'语句

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Channel Select 模拟输入通道选择
0	0	0	选择 P1.0 作为A/D输入来用
0	0	1	选择 P1.1 作为A/D输入来用
0	1	0	选择 P1.2 作为A/D输入来用
0	1	1	选择 P1.3 作为A/D输入来用
1	0	0	选择 P1.4 作为A/D输入来用
1	0	1	选择 P1.5 作为A/D输入来用
1	1	0	选择 P1.6 作为A/D输入来用
1	1	1	选择 P1.7 作为A/D输入来用

ADC_START: 模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换,转换结束后为0。

ADC_FLAG: 模数转换器转换结束标志位,当A/D转换完成后，ADC_FLAG = 1，要由软件清0。
不管是A/D转换完成后由该位申请产生中断，还是由软件查询该标志位A/D转换是否结束，
当A/D转换完成后，ADC_FLAG = 1，一定要软件清0。

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D转换所需时间
1	1	90个时钟周期转换一次，CPU工作频率21MHz时，A/D转换速度约300KHz
1	0	180个时钟周期转换一次
0	1	360个时钟周期转换一次
0	0	540个时钟周期转换一次

STC12C5S60AD/S2系列单片机的A/D转换模块说使用的时钟是外部晶体时钟或内部R/C振荡器所产生的系统时钟,不使用时钟分频寄存器CLK_DIV对系统时钟分频后所产生的供给CPU工作所使用的时钟.

好处:

这样可以让ADC用较高的频率工作,提高A/D的转换速度

这样可以让CPU用较低的频率工作,降低系统的功耗

程序中需要注意的事项:

由于是2套时钟,所以,设置ADC_CONTR控制寄存器后,要加4个空操作延时才可以正确读到ADC_CONTR寄存器的值,原因是设置ADC_CONTR控制寄存器的语句执行后,要经过4个CPU时钟的延时,其值才能够保证被设置进ADC_CONTR控制寄存器.

```
MOV  ADC_CONTR, #DATA
NOP
NOP
NOP
NOP
MOV  A,   ADC_CONTR    ;经过4个时钟延时后,才能够正确读到ADC_CONTR控制寄存器的值
```

ADC_POWER: ADC电源控制位。

0：关闭ADC电源；1：打开A/D转换器电源.建议进入空闲模式前，将ADC电源关闭，ADC_POWER =0.
启动AD转换前一定要确认AD电源已打开，AD转换结束后关闭AD电源可降低功耗，也可不关闭。
初次打开内部A/D转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动A/D转换
建议启动A/D转换后，在A/D转换结束之前，不改变任何I/O口的状态，有利于高精度A/D转换

ADC_RES特殊功能寄存器: A/D转换结果特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ADC_RES	BDh	A/D转换结果寄存器									0000,0000
ADC_RESL	BEh	A/D转换结果寄存器低									0000,0000
AUXR1	A2h	Auxiliary register 1		PCA_P4	SPI_P4	S2_P4	GF2	ADRJ	-	DPS	0000,00x0

AUXR1寄存器的ADRJ位是A/D转换结果寄存器(ADC_RES,ADC_RESL)的数据格式调整控制位

ADRJ: 0, 10位A/D转换结果的高8位存放在ADC_RES中，低2位存放在ADC_RESL的低2位中

Mnemonic	Name	7	6	5	4	3	2	1	0	Reset Value
ADC_RES	A/D转换结果寄存器	ADC_RES9	ADC_RES8	ADC_RES7	ADC_RES6	ADC_RES5	ADC_RES4	ADC_RES3	ADC_RES2	0000,0000
ADC_RESL	A/D转换结果寄存器低							ADC_RES1	ADC_RES0	0000,0000
AUXR1	Auxiliary register 1						ADRJ=0			

ADRJ: 1, 10位A/D转换结果的高2位存放在ADC_RES中低2位中，低8位存放在ADC_RESL中

Mnemonic	Name	7	6	5	4	3	2	1	0	Reset Value
ADC_RES	A/D转换结果寄存器							ADC_RES9	ADC_RES8	0000,0000
ADC_RESL	A/D转换结果寄存器低	ADC_RES7	ADC_RES6	ADC_RES5	ADC_RES4	ADC_RES3	ADC_RES2	ADC_RES1	ADC_RES0	0000,0000
AUXR1	Auxiliary register 1						ADRJ=1			

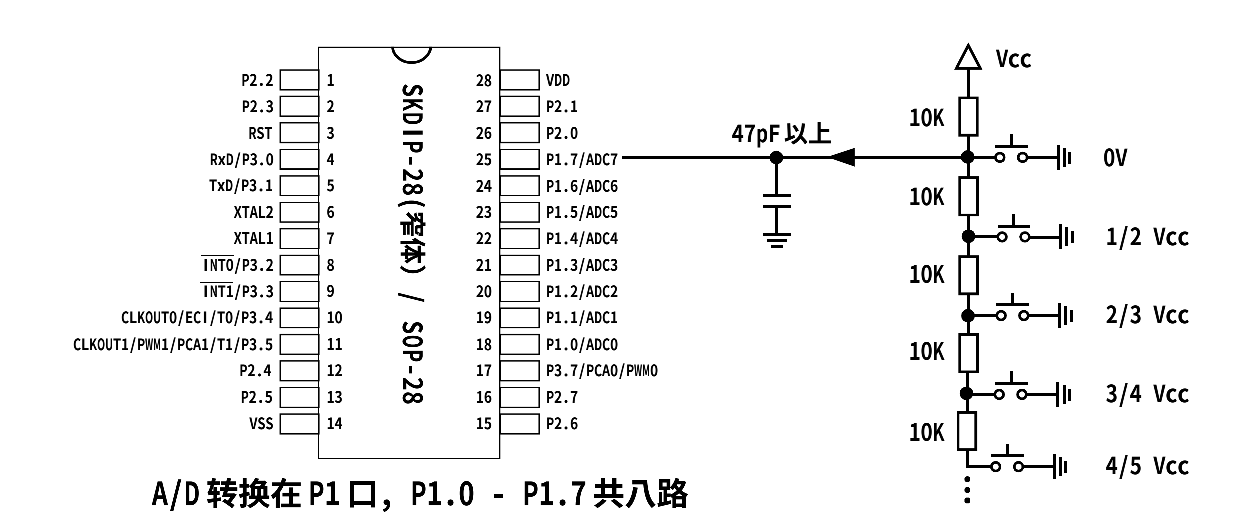
ADRJ = 0,模/数转换结果计算公式如下：取10位结果 (ADC_RES[7:0],ADC_RESL[1:0]) = 1024 x Vin / Vcc

ADRJ = 0,模/数转换结果计算公式如下：取8位结果 ADC_RES[7:0] = 256 x Vin / Vcc

ADRJ = 1,模/数转换结果计算公式如下：取10位结果 (ADC_RES[1:0],ADC_RESL[7:0]) = 1024 x Vin / Vcc

Vin为模拟输入通道输入电压，Vcc为单片机实际工作电压，用单片机工作电压作为模拟参考电压。

7.3 A/D转换典型应用线路，按键扫描



7.4 A/D转换模块的参考电压源

STC12C5201AD 系列单片机的参考电压源是输入工作电压 Vcc，所以一般不用外接参考电压源。如 7805 的输出电压是 5V，但实际电压可能是 4.88V 到 4.96V，用户需要精度比较高的话，可在出厂时将实际测出的工作电压值记录在单片机内部的 EEPROM 里面，以供计算。

如果有些用户的 Vcc 不固定，如电池供电，电池电压在 5.3V-4.2V 之间漂移，则 Vcc 不固定，就需要在 8 路 A/D 转换的一个通道外接一个稳定的参考电压源，来计算出此时的工作电压 Vcc，再计算出其他几路 A/D 转换通道的电压。

如可在 ADC 转换通道的第七通道外接一个 1.25V（或 1V，或...）的基准参考电压源，由此求出此时的工作电压 Vcc，再计算出其它几路 A/D 转换通道的电压。

7.5 一个完整的A/D转换测试程序

```
;/* ---  STC International Limited ------------------------------------------------ */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 -------------------------------------- */
;/* ---  演示 STC12C5201AD 系列 MCU 的 A/D 转换功能 ---------------------------------- */
;/* ---  Mobile: 13922805190 --------------------------------------------------------- */
;/* ---  Fax: 0755-82944243 ---------------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ---------------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ----------------------------------------------------- */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本程序用宏晶的 STC-ISP Ver 3.0A.PCB 的下载编程工具测试通过,相关的 A/D 转换结果在 P1 口上显示
;转换结果也以 16 进制形式输出到串行口，可以用串行口调试程序观察输出结果。
;时钟 18.432MHz，波特率 = 9600。
;转换结果也在 P1 口利用 LED 显示出来，方便观察。
LED_MCU_START   EQU  P3.7

ADC_CONTR   EQU  0BCH    ;A/D 转换寄存器
ADC_RES     EQU  0BDH    ;8 位 A/D 转换结果寄存器
P1ASF       EQU  9DH     ;P1 口中的相应位作为模拟功能使用时的控制寄存器,如做 A/D 用,相应位要置 1

ADC_Power_On_Speed_Channel_0 EQU 11100000B    ;P1.0 作为 A/D 输入
ADC_Power_On_Speed_Channel_1 EQU 11100001B    ;P1.1 作为 A/D 输入
ADC_Power_On_Speed_Channel_2 EQU 11100010B    ;P1.2 作为 A/D 输入
ADC_Power_On_Speed_Channel_3 EQU 11100011B    ;P1.3 作为 A/D 输入
ADC_Power_On_Speed_Channel_4 EQU 11100100B    ;P1.4 作为 A/D 输入
ADC_Power_On_Speed_Channel_5 EQU 11100101B    ;P1.5 作为 A/D 输入
ADC_Power_On_Speed_Channel_6 EQU 11100110B    ;P1.6 作为 A/D 输入
ADC_Power_On_Speed_Channel_7 EQU 11100111B    ;P1.7 作为 A/D 输入
;-----------------------------------------------
;定义变量
ADC_Channel_0_Result     EQU 30H    ;0 通道 A/D 转换结果
ADC_Channel_1_Result     EQU 31H    ;1 通道 A/D 转换结果
ADC_Channel_2_Result     EQU 32H    ;2 通道 A/D 转换结果
ADC_Channel_3_Result     EQU 33H    ;3 通道 A/D 转换结果
ADC_Channel_4_Result     EQU 34H    ;4 通道 A/D 转换结果
ADC_Channel_5_Result     EQU 35H    ;5 通道 A/D 转换结果
ADC_Channel_6_Result     EQU 36H    ;6 通道 A/D 转换结果
ADC_Channel_7_Result     EQU 37H    ;7 通道 A/D 转换结果
```

```
;------------------------------------------------
    ORG     0000H
    LJMP    MAIN

    ORG     0050H
MAIN:
    CLR     LED_MCU_START           ;MCU 工作指示灯 LED_MCU_START  EQU P3.7
    MOV     SP, #7FH                ;设置堆栈

    ACALL Initiate_RS232            ;初始化串口

    ACALL ADC_Power_On              ;开 ADC 电源，第一次使用时要打开内部模拟电源
                                    ;开 ADC 电源，可适当加延时，1mS 以内就足够了
    ACALL Set_P12_ASF               ;设置 P1.2 为模拟功能口
    ACALL Set_ADC_Channel_2         ;设置 P1.2 作为 A/D 转换通道

    ACALL Get_AD_Result             ;测量电压并且取 A/D 转换结果
    ACALL Send_AD_Result            ;发送转换结果到 PC 机

    ACALL Set_P12_Normal_IO         ;设置 P1.2 为普通 IO
    MOV     A, ADC_Channel_2_Result ;用 P1 口显示 A/D 转换结果
    CPL     A
    MOV     P1, A

Wait_Loop:
    SJMP    Wait_Loop               ;停机

;------------------------------------------------------------
;------------------------------------------------------------
;------------------------------------------------------------
;------------------------------------------------------------
Initiate_RS232:                     ;串口初始化
    CLR     ES                      ;禁止串口中断
    MOV     TMOD, #20H              ;设置 T1 为波特率发生器
    MOV     SCON, #50H              ;0101,0000 8 位数据位，无奇偶校验
    MOV     TH1, #0FBH              ;18.432MHz 晶振，波特率 = 9600
    MOV     TL1, #0FBH

    SETB    TR1                     ;启动 T1
    RET
;-------------------------------------------------------------
Send_Byte:
    CLR     TI
    MOV     SBUF, A
Send_Byte_Wait_Finish:
    JNB     TI, Send_Byte_Wait_Finish
    CLR     TI
    RET
```

```
;-------------------------------------------------------------
ADC_Power_On:
    PUSH  ACC
    ORL   ADC_CONTR, #80H           ;开 A/D 转换电源
    MOV   A, #20H
    ACALL Delay                     ;开 A/D 转换电源后要加延时，1mS 以内就足够了
    POP   ACC
    RET

;-------------------------------------------------------------
;设置 P1.2 为模拟功能
Set_P12_ASF:
    PUSH  ACC
    MOV   A, #00000100B
    ORL   P1ASF, A
    POP   ACC
    RET

;-------------------------------------------------------------
;设置 P1.2 为普通 IO
Set_P12_Normal_IO:
    PUSH  ACC
    MOV   A, #11111011B
    ANL   P1ASF, A
    POP   ACC
    RET

;-------------------------------------------------------------
Set_ADC_Channel_2:
    MOV   ADC_CONTR, #ADC_Power_On_Speed_Channel_2
                    ;选择 P1.2 作为 A/D 转换通道
    MOV   A, #05H   ;更换 A/D 转换通道后要适当延时, 使输入电压稳定
                    ;以后如果不更换 A/D 转换通道的话, 不需要加延时
    ACALL Delay     ;切换 A/D 转换通道,加延时 20uS～200uS 就可以了,与输入电压源的内阻有关
                    ;如果输入电压信号源的内阻在 10K 以下,可不加延时
    RET

;-------------------------------------------------------------

Send_AD_Result:
    PUSH  ACC
    MOV   A, ADC_Channel_2_Result   ;取 AD 转换结果
    ACALL Send_Byte                 ;发送转换结果到 PC 机
    POP   ACC
    RET
```

```
;-----------------------------------------------------------
Get_AD_Result:
    PUSH  ACC                      ;入栈保护
    MOV   ADC_RES, #0
    ORL   ADC_CONTR, #00001000B    ;启动 AD 转换
    NOP   ;在对 ADC_CONTR 寄存器进行写操作后,要加 4 个空操作延时,才能够正确读到 ADC_CONTR 的值
    NOP   ;在对 ADC_CONTR 寄存器进行写操作后,要加 4 个空操作延时,才能够正确读到 ADC_CONTR 的值
    NOP   ;在对 ADC_CONTR 寄存器进行写操作后,要加 4 个空操作延时,才能够正确读到 ADC_CONTR 的值
    NOP   ;在对 ADC_CONTR 寄存器进行写操作后,要加 4 个空操作延时,才能够正确读到 ADC_CONTR 的值
Wait_AD_Finishe:
    MOV   A, #00010000B            ;判断 AD 转换是否完成
    ANL   A, ADC_CONTR
    JZ    Wait_AD_Finishe          ;AD 转换尚未完成，继续等待

    ANL   ADC_CONTR, #11100111B    ;清 0 ADC_FLAG, ADC_START 位，停止 A/D 转换

    MOV   A, ADC_RES
    MOV   ADC_Channel_2_Result, A  ;保存 AD 转换结果
    POP   ACC
    RET
;-----------------------------------------------------------------
Delay:
    PUSH  02                       ;将寄存器组 0 的 R2 入栈
    PUSH  03                       ;将寄存器组 0 的 R3 入栈
    PUSH  04                       ;将寄存器组 0 的 R4 入栈
    MOV   R4, A
Delay_Loop0:
    MOV   R3, #200             ;2 CLOCK  ---------------------+
Delay_Loop1:                   ;                              |
    MOV   R2, #249             ;2 CLOCK  ------+              |
Delay_Loop:                    ;               | 1002 CLOCK   |200406 CLOCK
    DJNZ  R2, Delay_Loop       ;4 CLOCK        |              |
    DJNZ  R3, Delay_Loop1      ;4 CLOCK  ------+              |
    DJNZ  R4, Delay_Loop0      ;4 CLOCK  ---------------------+

    POP   04
    POP   03
    POP   02
    RET
;------------------------------------------------------------------
    END
```

第八章　STC12 系列单片机的 PCA/PWM 应用

8.1　PCA/PWM寄存器列表

STC12C5201AD 系列 1T 8051 单片机 PCA/PWM 特殊功能寄存器表　PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF	0xxx,0000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00

CMOD - PCA 模式寄存器（地址：D9H）

位	7	6	5	4	3	2	1	0
符 号	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF

CMOD - PCA 模式 寄存器的位描述　（地址：D9H）

位	符号	描述
7	CIDL	计数器阵列空闲控制：CIDL=0时，空闲模式下PCA计数器继续工作。CIDL＝1时，空闲模式下PCA计数器停止工作。
6 - 4	-	保留为将来之用。
3- 1	CPS2,CPS1,CPS0	PCA计数脉冲选择（见下表）。
0	ECF	PCA计数溢出中断使能：ECF=1时，使能寄存器CCON CF位的中断。ECF=0时，禁止该功能。

CMOD - PCA 计数器阵列的计数脉冲选择　（地址：D9H）

CPS2	CPS1	CPS0	选择PCA/PWM时钟源输入
0	0	0	0，系统时钟，Fosc/12
0	0	1	1，系统时钟，Fosc/2
0	1	0	2，定时器0的溢出,可以实现可调频率的PWM输出
0	1	1	3，ECI/P3.4脚的外部时钟输入（最大速率＝Fosc/2）
1	0	0	4，系统时钟，Fosc
1	0	1	5，系统时钟/4，Fosc/4
1	1	0	6，系统时钟/6，Fosc/6
1	1	1	7，系统时钟/8，Fosc/8

CPS2/CPS1/CPS0 = 1/0/0 时，PCA/PWM 的时钟源是 Fosc，不用 Timer0,PWM 的频率为 Fosc/256
如果要得到系统时钟 / 3 来作为 PCA 的时钟源，应让 T0 工作在 1T 模式，记数 3 个脉冲即产生溢出。
如果此时使用内部 RC 作为系统时钟(室温情况下，5V 单片机为 11MHz~15.5MHz)，可以输出 14K~19K 频率的 PWM。用 T0 的溢出可对系统时钟进行 1~256 级分频

CCON - PCA 控制寄存器的位分配 （地址：D8H）

位	7	6	5	4	3	2	1	0
符 号	CF	CR	-	-	-	-	CCF1	CCF0

CCON - PCA 控制寄存器的位描述 （地址：D8H）

位	符号	描述
7	CF	PCA计数器阵列溢出标志。计数值翻转时该位由硬件置位。如果CMOD寄存器的ECF位置位，CF标志可用来产生中断。CF位可通过硬件或软件置位，但只可通过软件清零。
6	CR	PCA计数器阵列运行控制位。该位通过软件置位，用来起动PCA计数器阵列计数。该位通过软件清零，用来关闭PCA计数器。
5 - 4	-	保留位，保留为将来使用。
3	-	保留位，保留为将来使用。
2	-	保留位，保留为将来使用。
1	CCF1	PCA模块1中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。
0	CCF0	PCA模块0中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。

CCAPMn - PCA 比较 / 捕获模块寄存器的位分配（CCAPM0 地址：0DAH；CCAPM1 地址：0DBH）

位	7	6	5	4	3	2	1	0
符 号	-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

CCAPMn - PCA 比较 / 捕获模块寄存器的位描述 （ n：0,1 ）

位	符号	描述 n : 0,1
7	-	保留为将来之用。
6	ECOMn	使能比较器。ECOMn＝1时使能比较器功能。
5	CAPPn	正捕获。CAPPn＝1时使能上升沿捕获。
4	CAPNn	负捕获。CAPNn＝1时使能下降沿捕获。
3	MATn	匹配。当MATn＝1时，PCA计数值与模块的比较/捕获寄存器的值的匹配将置位CCON寄存器的中断标志位CCFn。
2	TOGn	翻转。当TOGn＝1时，工作在PCA高速输出模式，PCA计数器的值与模块的比较/捕获寄存器的值的匹配将使CEXn脚翻转。(CEX0/PCA0/PWM0/P3.7,CEX1/PCA0/PWM0/P3.5)
1	PWMn	脉宽调节模式。当PWMn＝1时，使能CEXn脚用作脉宽调节输出。
0	ECCFn	使能CCFn中断。使能寄存器CCON的比较/捕获标志CCFn，用来产生中断。

PCA 模块工作模式（CCAPMn 寄存器，n：0,1）

-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
	0	0	0	0	0	0	0	无此操作
	1	0	0	0	0	1	0	8位PWM,无中断
	1	1	0	0	0	1	1	8位PWM输出，由低变高可产生中断
	1	0	1	0	0	1	1	8位PWM输出，由高变低可产生中断
	1	1	1	0	0	1	1	8位PWM输出，由低变高或者由高变低均可产生中断
	X	1	0	0	0	0	X	16位捕获模式，由CEXn/PCAn的上升沿触发
	X	0	1	0	0	0	X	16位捕获模式，由CEXn/PCAn的下降沿触发
	X	1	1	0	0	0	X	16位捕获模式，由CEXn/PCAn的跳变触发
	1	0	0	1	0	0	X	16位软件定时器
	1	0	0	1	1	0	X	16位高速输出

8.2　PCA/PWM功能介绍

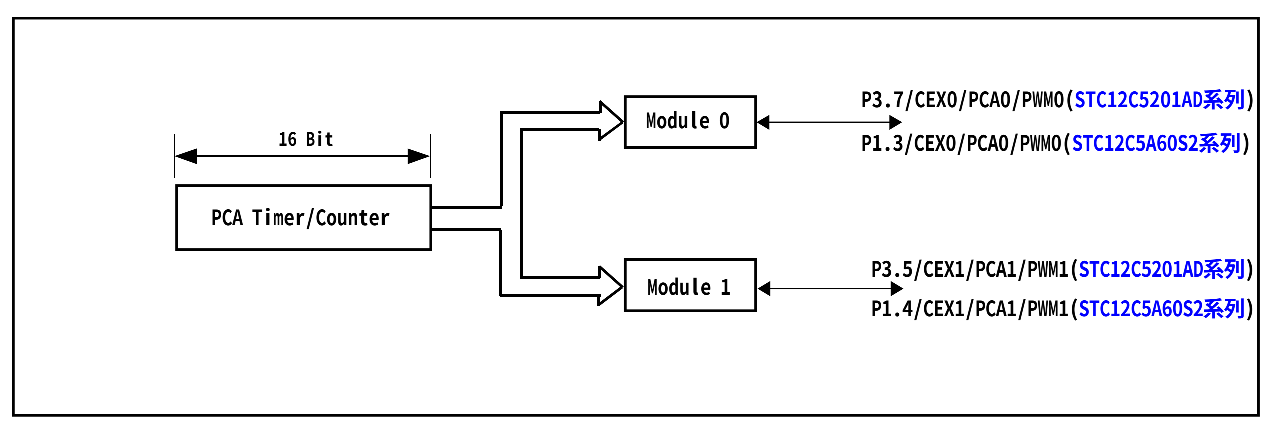
STC12xx 系列单片机有 2 路可编程计数器阵列 PCA/PWM(其中 STC12C5A60S2 系列单片机可以通过 AUXR1 寄存器设置 PCA/PWM 从 P1 口切换到 P4 口)，详情请参阅 P4 口的使用相关章节内容。

PCA 含有一个特殊的 16 位定时器，有 2 个 16 位的捕获 / 比较模块与之相连。每个模块可编程工作在 4 种模式下：上升 / 下降沿捕获、软件定时器、高速输出或可调制脉冲输出。

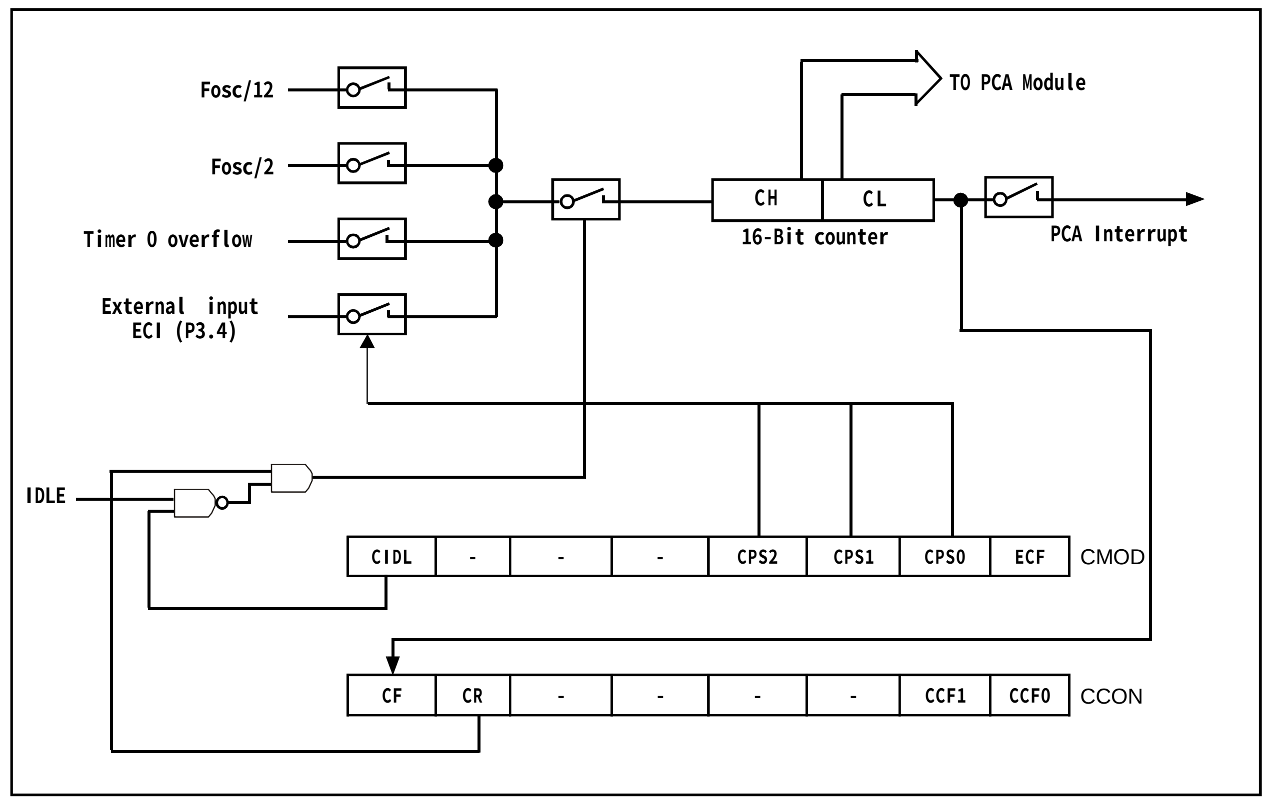
STC12C5201AD 系列: 模块 0 连接到 P3.7/CCP0,模块 1 连接到 P3.5/CCP1。

STC12C5A60S2 系列: 模块 0 连接到 P1.3/CCP0(可以切换到 P4.2/CCP0/MISO 口),
模块 1 连接到 P1.4/CCP1(可以切换到 P4.3/CCP1/SCLK 口)。

寄存器 CH 和 CL 的内容是正在自由递增计数的 16 位 PCA 定时器的值。PCA 定时器是 2 个模块的公共时间基准，可通过编程工作在：1/12 振荡频率、1/8 振荡频率、1/6 振荡频率、1/4 振荡频率、1/2 振荡频率、振荡频率、定时器 0 溢出或 ECI 脚的输入（STC12C5201AD 系列在 P3.4 口，STC12C5A60S2 系列在 P1.2 口)。定时器的计数源由 CMOD SFR 的 CPS2,CPS1 和 CPS0 位来确定（见 CMOD 特殊功能寄存器说明）。



Programmable　Counter　Array



PCA　Timer/Counter

CMOD　SFR 还有 2 个位与 PCA 相关。它们分别是：CIDL，空闲模式下允许停止 PCA；ECF，置位时，使能 PCA 中断，当 PCA 定时器溢出将 PCA 计数溢出标志 CF（CCON　SFR）置位。

CCON　SFR 包含 PCA 的运行控制位（CR）和 PCA 定时器标志（CF）以及各个模块的标志（CCF1/CCF0）。通过软件置位 CR 位（CCON.6）来运行 PCA。CR 位被清零时 PCA 关闭。当 PCA 计数器溢出时，CF 位（CCON.7）置位,如果 CMOD 寄存器的 ECF 位置位,就产生中断。CF 位只可通过软件清除。CCON 寄存器的位 0～3 是 PCA 各个模块的标志（位 0 对应模块 0，位 1 对应模块 1），当发生匹配或比较时由硬件置位。这些标志也只能通过软件清除。所有模块共用一个中断向量。PCA 的中断系统如图所示。

PCA 的每个模块都对应一个特殊功能寄存器。它们分别是：模块 0 对应 CCAPM0，模块 1 对应 CCAPM1，特殊功能寄存器包含了相应模块的工作模式控制位。

当模块发生匹配或比较时，ECCFn 位（CCAPMn.0，n ＝ 0，1 由工作的模块决定）使能 CCON　SFR 的 CCFn 标志来产生中断。

PWM（CCAPMn.1）用来使能脉宽调制模式。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果 TOG 位（CCAPMn.2）置位，模块的 CEXn 输出将发生翻转。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果匹配位 MATn（CCAPMn.3）置位，CCON 寄存器的 CCFn 位将被置位。

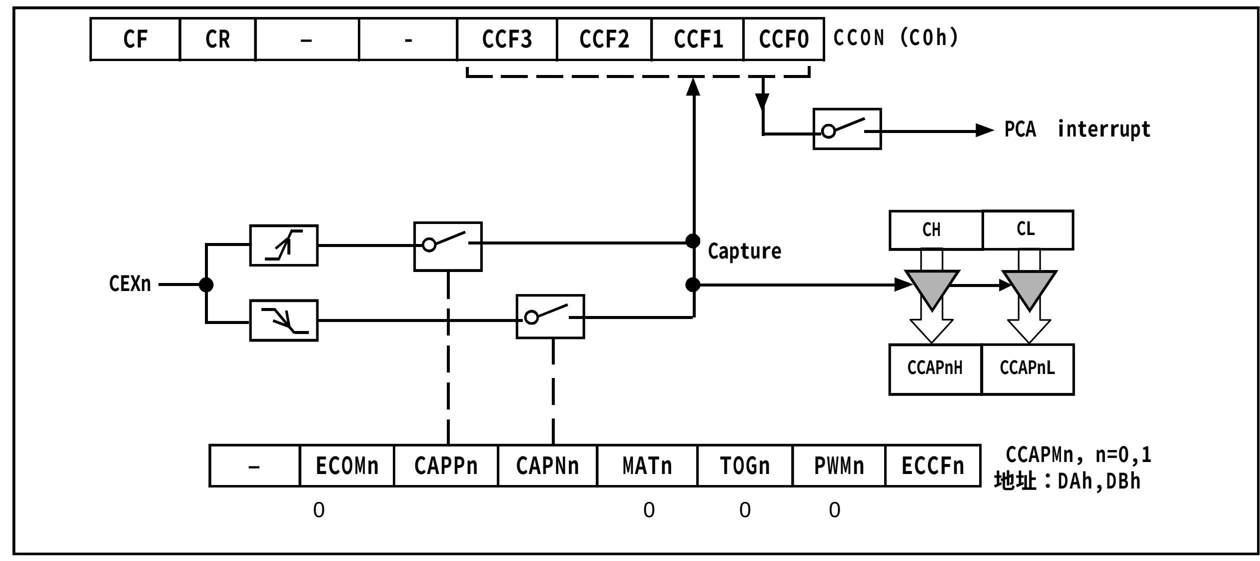
CAPNn（CCAPMn.4）和 CAPPn（CCAPMn.5）用来设置捕获输入的有效沿。CAPNn 位使能下降沿有效，CAPPn 位使能上升沿有效。如果两位都置位，则两种跳变沿都被使能，捕获可在两种跳变沿产生。

通过置位 CCAPMn 寄存器的 ECOMn 位（CCAPMn.6）来使能比较器功能。

每个 PCA 模块还对应另外两个寄存器，CCAPnH 和 CCAPnL。当出现捕获或比较时，它们用来保存 16 位的计数值。当 PCA 模块用在 PWM 模式中时，它们用来控制输出的占空比。

PCA 捕获模式

要使一个 PCA 模块工作在捕获模式（下图），寄存器 CCAPMn 的两位（CAPNn 和 CAPPn）或其中任何一位必须置 1。对模块的外部 CEXn 输入（STC12C5201AD 系列：CEX0/P3.7，CEX1/P3.5,STC12C5A60S2 系列 CEX0/P1.3,CEX1/P1.4）的跳变进行采样。当采样到有效跳变时，PCA 硬件就将 PCA 计数器阵列寄存器（CH 和 CL）的值装载到模块的捕获寄存器中（CCAPnL 和 CCAPnH)。

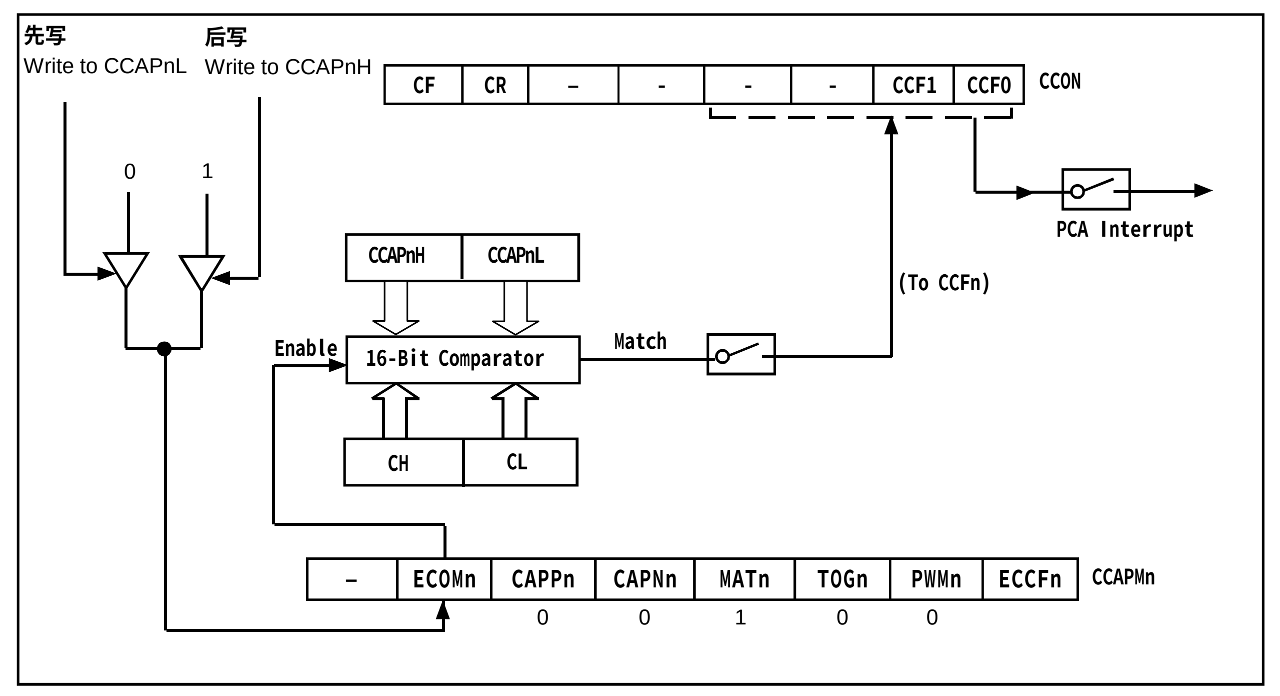


PCA Capture Mode （PCA 捕获模式图）

如果 CCON SFR 的位 CCFn 和 CCAPMn SFR 的位 ECCFn 位被置位，将产生中断。

16 位软件定时器模式

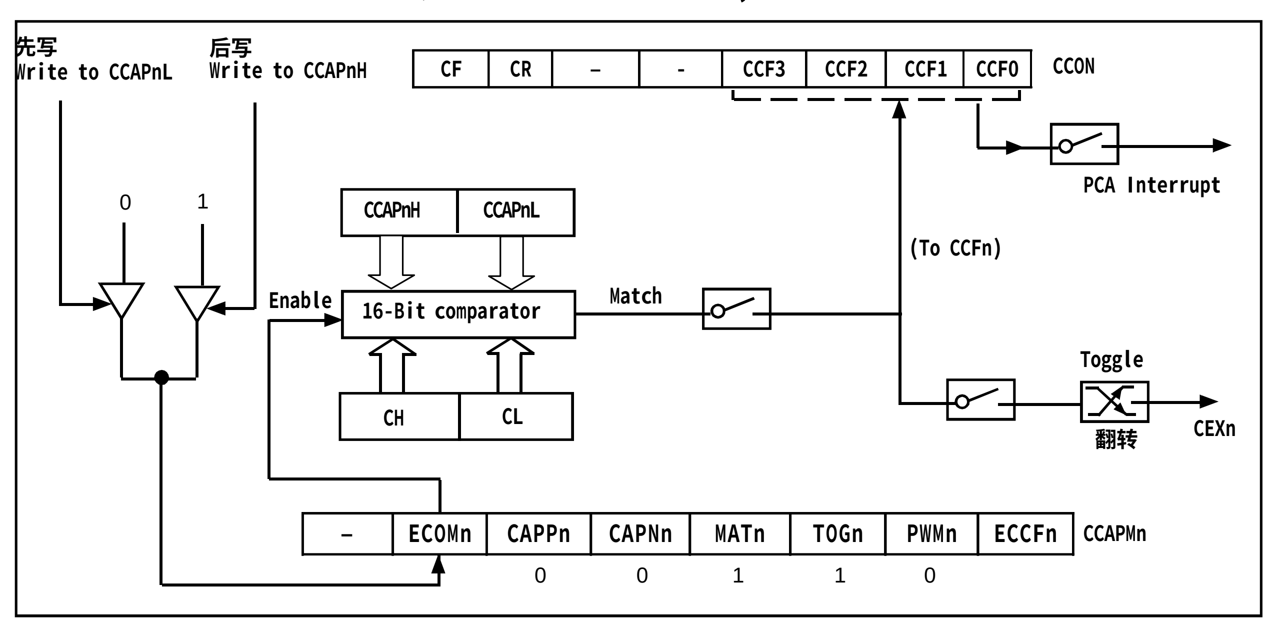
通过置位 CCAPMn 寄存器的 ECOM 和 MAT 位，可使 PCA 模块用作软件定时器（下图）。PCA 定时器的值与模块捕获寄存器的值相比较，当两者相等时，如果位 CCFn（在 CCON SFR 中）和位 ECCFn（在 CCAPMn SFR 中）都置位，将产生中断。



PCA Software Timer Mode/ 软件定时器模式 /PCA 比较模式

高速输出模式

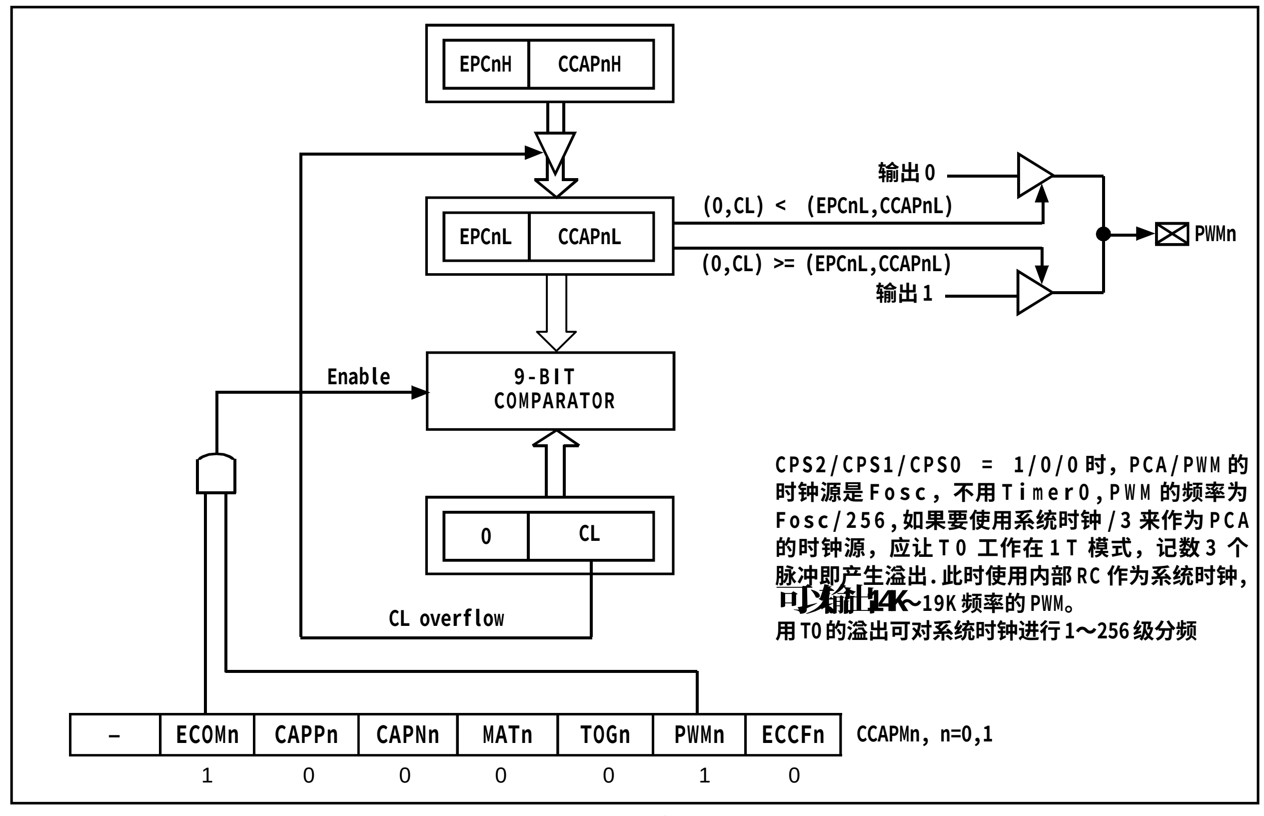
该模式中（下图），当 PCA 计数器的计数值与模块捕获寄存器的值相匹配时，PCA 模块的 CEXn 输出将发生翻转。要激活高速输出模式，模块 CCAPMn SFR 的 TOG,MAT 和 ECOM 位必须都置位。



PCA High-Speed Output Mode / PCA 高速输出模式

脉宽调节模式(PWM)

所有 PCA 模块都可用作 PWM 输出（下图）。输出频率取决于 PCA 定时器的时钟源。



PCA PWM mode / 可调制脉冲宽度输出模式

由于所有模块共用仅有的 PCA 定时器，所有它们的输出频率相同。各个模块的输出占空比是独立变化的，与使用的捕获寄存器｛EPCnL，CCAPnL｝有关。当 CL SFR 的值小于｛EPCnL，CCAPnL｝时，输出为低，当 PCA CL SFR 的值等于或大于｛EPCnL，CCAPnL｝时，输出为高。当 CL 的值由 FF 变为 00 溢出时，｛EPCnH，CCAPnH｝的内容装载到｛EPCnL，CCAPnL｝中。这样就可实现无干扰地更新 PWM。要使能 PWM 模式，模块 CCAPMn 寄存器的 PWMn 和 ECOMn 位必须置位。

由于 PWM 是 8 位的，所以：$\text{PWM 的频率} = \dfrac{\text{PCA 时钟输入源频率}}{256}$

PCA 时钟输入源可以从以下 4 种中选择一种：

Fosc， Fosc/2， Fosc/4，Fosc/6，Fosc/8，Fosc/12,定时器 0 的溢出，ECI/P3.4 输入

举例：要求 PWM 输出频率为 38KHz，选 Fosc 为 PCA/PWM 时钟输入源，求出 Fosc 的值

由计算公式 38000 = Fosc / 256 ，得到外部时钟频率 Fosc = 38000 x 256 x 1 =9,728,000

如果要实现可调频率的 PWM 输出,可选择定时器 0 的溢出率或者 ECI 脚的输入作为 PCA/PWM 的时钟输入源

当 EPCnL = 0 及 ECCAPnL = 00H 时,PWM 固定输出高

当 EPCnL = 1 及 CCAPnL = 0FFH 时,PWM 固定输出低

当某个 I/O 口作为 PWM 使用时,该口的状态:

PWM之前口的状态	PWM输出时口的状态
弱上拉 / 准双向口	强推挽输出 / 强上拉输出 ，要加输出限流电阻1K-10K
强推挽输出 / 强上拉输出	强推挽输出 / 强上拉输出 ，要加输出限流电阻1K-10K
仅为输入 / 高阻	PWM无效
开漏	开漏

限流电阻用 10K 到 1K

普通 I/O 口 —— 接负载

8.3 用PCA 功能扩展外部中断的示例程序

```
;/* --- STC International Limited ---------------------------------------------------- */
;/* --- 宏晶科技    姚永平    2006/1/6    V1.0 ----------------------------------------- */
;/* --- 使用 STC12C5201AD/STC12C5A60S2 系列单片机 PCA 功能扩展外部中断的示例程序 ------ */
;/* --- Mobile: 13922805190 ----------------------------------------------------------- */
;/* --- Fax: 0755-82944243 ------------------------------------------------------------ */
;/* --- Tel: 0755-82948409 ------------------------------------------------------------ */
;/* --- Web: www.mcu-memory.com ------------------------------------------------------- */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本演示程序在STC-ISP Ver 3.0A.PCB的下载编程工具上测试通过
;-------------------------------------------------------------------------------
;P3.7(PCA 模块0) 扩展为下降沿外部中断,
;P3.5(PCA 模块1) 扩展为上升沿/下降沿都可触发的外部中断。
;
;1) 汇编源程序，把汇编程序产生的的程序代码下载到单片机中，上电运行本程序。
;2) 将 P3.7/PCA0 短路到地，这一动作产生一个下降沿，此时本演示程序对 P1.6 取反，
;    P1.6 控制的 LED 灯将会变化一次。
;3) 改变 P3.5/PCA1 的外部高低状态(由高到低--产生下降沿；由低到高--产生上升沿)，
;    本演示程序在 P3.5/PCA1 的下降沿/上升沿都产生中断，此时本演示程序对 P1.5 取反，
;    P1.5 控制的 LED 灯状态将会发生变化。
;    所谓 LED 灯状态发生变化是指 LED 由灭变亮或由亮变灭。
;-------------------------------------------------------------------------------
;声明STC12C5201AD系列 MCU 特殊功能寄存器地址
IPH     EQU  0B7H          ;中断优先级高位寄存器
CH      EQU  0F9H          ;PCA 计数器高8位。
CL      EQU  0E9H          ;PCA 计数器低8位。
;-----------------------
CCON    EQU  0D8H          ;PCA 控制寄存器。
CCF0    EQU  CCON.0        ;PCA 模块0 中断标志，由硬件置位，必须由软件清0。
CCF1    EQU  CCON.1        ;PCA 模块1 中断标志，由硬件置位，必须由软件清0。
CR      EQU  CCON.6        ;1:允许 PCA 计数器计数，必须由软件清0。
CF      EQU  CCON.7        ;PCA计数器溢出(CH,CL由 FFFFH 变为 0000H)标志,
                           ;PCA计数器溢出后由硬件置位,必须由软件清0。
```

```
;------------------------
CMOD      EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数, 0: 继续计数, 1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1   CPS0
;             0      0     外部晶体频率 /12。
;             0      1     外部晶体频率 /2。
;             1      0     Timer 0 溢出脉冲,
;                          Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;             1      1     从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0     ECF: PCA 计数器溢出中断允许位, 1-- 允许 CF(CCON.7) 产生中断。
;------------------------

CCAP0H    EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP0L    EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。

;------------------------
PCA_PWM0 EQU    0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU    0F3H              ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下, 与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下, 与 CCAPnL 组成 9 位数。
```

```
;------------------------
CCAPM0    EQU    0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1    EQU    0DBH              ;PCA 模块 1 的工作模式寄存器。

;CCAPMn:     7      6      5      4      3      2      1      0
;            -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0      0     0     0     0    00H 未启用任何功能。
;  x      1      0      0     0     0     x    21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1      0     0     0     x    11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1      0     0     0     x    31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0      1     0     0     x    49H 16 位软件定时器。
;  1      0      0      1     1     0     x    4DH 16 位高速脉冲输出。
;  1      0      0      0     0     1     0    42H 8 位 PWM。
;-------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_PCA_INT0        EQU P1.6
LED_PCA_INT1        EQU P1.5
;-------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-------------------------------------------------------------------------------
    ORG   003BH                  ;interrupt 7(0,1,2,3,4,5,6,7)
    LJMP  PCA_Interrupt
;-------------------------------------------------------------------------------
    ORG   0050H
MAIN:
    MOV   SP, #7FH
    CLR   LED_MCU_START          ;点亮 LED_MCU_START LED，表示程序正在运行
    LCALL PCA_Initiate           ;初始化 PCA
WAIT:
    SJMP  WAIT                   ;跳转到本行，无限循环。
```

```
;-----------------------------------------------------------------------------
PCA_Initiate:
    MOV   CMOD, #10000000B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 fosc/12
                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H        ;CF = 0, 清 0 PCA 计数器溢出中断请求标志位
                           ;CR = 0, 不允许 PCA 计数器计数
                            ;清 0 PCA 各模块中断请求标志位, 如 CCF1, CCF0
    MOV   CL, #00H         ;清 0 PCA 计数器
    MOV   CH, #00H
;-------------------------
;设置模块 0
    MOV   CCAPM0, #11H ;设置 PCA 模块 0 下降沿触发捕捉功能,ECCF0 = 1,允许产生中断
;   MOV   CCAPM0, #21H ;如果送的是 #21h, 则 PCA 模块 0 为上升沿触发,ECCF0 = 1,允许产生中断
;-------------------------
;设置模块 1
    MOV   CCAPM1, #31H ;设置 PCA 模块 1 上升沿 / 下降沿均可触发的捕捉功能,ECCF1 = 1,可产生中断
;-------------------------
    SETB  EA                    ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                    ;启动 PCA 计数器(CH,CL)计数
    RET
;-----------------------------------------------------------------------------
PCA_Interrupt:
    PUSH  ACC
    PUSH  PSW

    JNB   CCF0, Not_PCA0_Else_PCA1 ;如果 CCF0 不等于 1 就不是 PCA 模块 0 中断
                                   ;就直接去判是否是 PCA 模块 1 中断
    ;模块 0 中断服务程序
    CPL   LED_PCA_INT0             ;P1.6 LED 变化一次, 表示 PCA 模块 0 发生了一次中断
    CLR   CCF0                     ;清 PCA 模块 0 中断标志

Not_PCA0_Else_PCA1:
    JNB   CCF1, PCA_Interrupt_Exit ;如果 CCF1 不等于 1 就不是 PCA 模块 1 中断
                                   ;就立即退出
    ;模块 1 中断服务程序
    CPL   LED_PCA_INT1             ;P1.5 LED 变化一次, 表示 PCA 模块 1 发生了一次中断
    CLR   CCF1                     ;清 PCA 模块 1 中断标志

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;-----------------------------------------------------------------------------
    END
;-----------------------------------------------------------------------------
```

8.4 用PCA 功能做定时器的示例程序

```
;/* --- STC International Limited ---------------------------------------------------- */
;/* --- 宏晶科技    姚永平    2006/1/6    V1.0 -------------------------------------- */
;/* --- PCA_12C5201_ASM_Timer ------------------------------------------------------------ */
;/* --- 使用 STC12C5201AD 系列单片机 PCA 功能做定时器的示例程序 ----------------------- */
;/* --- STC12C5201AD, STC12C5202AD, STC12C5203AD ------------------------------------ */
;/* --- STC12C5204AD, STC12C5206AD, STC12C5206AD ------------------------------------ */
;/* --- Mobile: 13922805190 ------------------------------------------------------------ */
;/* --- Fax: 0755-82944243 ------------------------------------------------------------- */
;/* --- Tel: 0755-82948409 ------------------------------------------------------------- */
;/* --- Web: www.mcu-memory.com -------------------------------------------------------- */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过
;-----------------------------------------------------------------------------
;晶振频率 Fosc = 18.432MHz，在 P1.5 输出脉冲宽度为 1 秒钟的方波
;-----------------------------------------------------------------------------
;声明 STC12C5201AD 系列 MCU 特殊功能寄存器地址
IPH     EQU    0B7H          ;中断优先级高位寄存器
CH      EQU    0F9H          ;PCA 计数器高 8 位。
CL      EQU    0E9H          ;PCA 计数器低 8 位。
;-----------------------
CCON    EQU    0D8H          ;PCA 控制寄存器。
CCF0    EQU    CCON.0        ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1    EQU    CCON.1        ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CR      EQU    CCON.6        ;1:允许 PCA 计数器计数，必须由软件清 0。
CF      EQU    CCON.7        ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                             ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
;-----------------------
CMOD    EQU    0D9H          ;PCA 工作模式寄存器。
;CMOD.7    CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。
;CMOD.2        PS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1        CPS0: PCA 计数器计数脉冲源选择位 0。
;CMOD.1        CPS0: PCA 计数器计数脉冲源选择位 0。
;         CPS2  CPS1  CPS0
;         0     0     0     系统时钟频率 /12
;         0     0     1     系统时钟频率 /2
;         0     1     0     Timer 0 溢出脉冲
;         0     1     1     ECI/P3.4 脚的外部时钟输入，最大速率 = Fosc/2
;         1     0     0     系统时钟频率
;         1     0     1     系统时钟频率 /4
;         1     1     0     系统时钟频率 /6
;         1     1     1     系统时钟频率 /8
;CMOD.0    ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
```

```
;------------------------
CCAP0H   EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H   EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。

CCAP0L   EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L   EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。


;------------------------
PCA_PWM0 EQU   0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU   0F3H              ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;------------------------
CCAPM0   EQU   0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH              ;PCA 模块 1 的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
```

```
;ECOMn   CAPPn   CAPNn   MATn   TOGn   PWMn   ECCFn
;  0       0       0       0      0      0      0     00H, 未启用任何功能。
;  1       0       0       0      0      1      0    42H, 8位 PWM
;  1       1       0       0      0      1      1    63H, 8位 PWM,由低变高可产生中断,上升沿中断
;  1       0       1       0      0      1      1    53H, 8位 PWM,由高变低可产生中断,下降沿中断
;  1       1       1       0      0      1      1    73H, 8位 PWM,由低变高和由高变低均可产生中断
;  x       1       0       0      0      0      x    21H, 16位CEXn上升沿触发捕捉功能
;  x       0       1       0      0      0      x    11H, 16位CEXn下降沿触发捕捉功能
;  x       1       1       0      0      0      x    31H, 16位CEXn边沿(上、下沿)触发捕捉功能
;  1       0       0       1      0      0      x    49H, 16位软件定时器
;  1       0       0       1      1      0      x    4DH, 16位高速脉冲输出
;-----------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_5mS_Flashing    EQU P1.6
LED_1S_Flashing     EQU P1.5
;-----------------------------------------------------------------------------
;定义常量
;Channe0_5mS_H, Channe0_5mS_L 的计算方法见 PCA 中断服务程序内的注释
Channe0_5mS_H       EQU 1EH      ;模块0 5mS 定时常数高位
Channe0_5mS_L       EQU 00H      ;模块0 5mS 定时常数低位
;-----------------------------------------------------------------------------
;定义变量
Counter              EQU 30H      ;声明一个计数器，用来计数中断的次数
;-----------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
    ORG   003BH                  ;interrupt 7(0,1,2,3,4,5,6,7)
    LJMP  PCA_interrupt
;-----------------------------------------------------------------------------
    ORG   0050H
MAIN:
    CLR   LED_MCU_START          ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    MOV   Counter, #0            ;清Counter计数器
    ACALL PCA_Initiate           ;初始化 PCA
WAIT:
    SJMP  WAIT                   ;跳转到本行，无限循环。
;-----------------------------------------------------------------------------
PCA_Initiate:
    MOV   CMOD, #10000000B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 fosc/12
                           ;禁止 PCA 计数器溢出(CH,CL由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H       ;CF = 0, 清0 PCA 计数器溢出中断请求标志位
                           ;CR = 0, 不允许 PCA 计数器计数
                           ;清0 PCA 各模块中断请求标志位，如 CCF1, CCF0
    MOV   CL, #00H         ;清0 PCA 计数器
    MOV   CH, #00H
```

```
;------------------------
    ;Channe0_5mS_H, Channe0_5mS_L 的计算方法见 PCA 中断服务程序内的注释
    MOV   CCAP0L, #Channe0_5mS_L ;给 PCA 模块 0 的 CCAP0L 置初值
    MOV   CCAP0H, #Channe0_5mS_H ;给 PCA 模块 0 的 CCAP0H 置初值
    MOV   CCAPM0, #49H      ;设置 PCA 模块 0 为 16 位软件定时器,ECCF0=1 允许 PCA 模块 0 中断
     ;当[CH, CL]==[CCAP0H, CCAP0L]时，产生中断请求，CCF0=1，请求中断
    SETB  EA                    ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                    ;启动 PCA 计数器(CH,CL)计数
    RET
;-------------------------------------------------------------------------
PCA_Interrupt:
    PUSH  ACC
    PUSH  PSW

    CPL   LED_5mS_Flashing   ;本程序 PCA 模块 0 每 5mS 中断一次，每次进中断将该灯状态取反

    ;在本程序中[CH,CL]每 12 个时钟脉冲加 1，当[CH,CL] 增加到等于 [CCAP0H，CCAP0L]时
    ;CCF0=1，产生中断请求。如果每次 PCA 模块 0 中断后，在中断服务程序中给
    ;[CCAP0H，CCAP0L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe0_5mS_H, Channe0_5mS_L

    ;举例: 时钟频率 Fosc = 18.432MHz, PCA 计数器计数 1E00H 次才是 5mS。
    ;计算 PCA 计数器计数多少次：
    ;     Channe0_5mS_H, Channe0_5mS_L = T/( (1/Fosc)*12 )
    ;                                  = 0.005/ ( (1/18432000)*12 )
    ;                                  = 7680 (10 进制数)
    ;                                  = 1E00H (16 进制数)
    ;     即 Channe0_5mS_H = 1EH, Channe0_5mS_L = 00H
    ;
    ;    Channe0_5mS_H, Channe0_5mS_L：每次给 [CCAP0H,CCAP0L] 增加的数值(步长)

    MOV   A, #Channe0_5mS_L      ;给[CCAP0H，CCAP0L] 增加一个数值
    ADD   A, CCAP0L
    MOV   CCAP0L, A
    MOV   A, #Channe0_5mS_H
    ADDC  A, CCAP0H
    MOV   CCAP0H, A
    CLR   CCF0                  ;清 PCA 模块 0 中断标志

    INC   Counter               ;中断次数计数器 + 1
    MOV   A, Counter
    CLR   C
    SUBB  A, #200                ;检测是否中断了 200 次 (1 秒)
    JC    PCA_Interrupt_Exit    ;有借位，表示 Counter 小于 200，立即跳转退出
```

```
    MOV   Counter, #0          ;已中断了 200 次，清 0 中断次数计数器
    CPL   LED_1S_Flashing      ;在 LED_1S_Flashing 输出脉冲宽度为 1 秒钟的方波

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;------------------------------------------------------------------------------
    END
```

8.5 PWM输出C语言示例程序

```
/* --- STC International Limited ---------------------------------------- */
/* --- 宏晶科技     姚永平     2006/1/6     V1.0 --------------------------- */
/* --- 使用 STC12C5201AD 系列单片机 PWM 输出 C 语言示例程序 ----------------- */
/* --- Mobile: 13922805190 ---------------------------------------------- */
/* --- Fax: 0755-82944243 ----------------------------------------------- */
/* --- Tel: 0755-82948409 ----------------------------------------------- */
/* --- Web: www.mcu-memory.com ------------------------------------------ */
/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ----------- */
/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 -----*/
/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 -----*/

#include<reg52.h>
sfr  CCON   = 0xD8;
sfr  CMOD   = 0xD9;
sfr  CL     = 0xE9;
sfr  CH     = 0xF9;
sfr  CCAP0L = 0xEA;
sfr  CCAP0H = 0xFA;
sfr  CCAPM0 = 0xDA;
sfr  CCAPM1 = 0xDB;
sbit  CR = 0xDE;
void main(void)
{
    CMOD = 0x02; // Setup PCA timer
    CL = 0x00;
    CH = 0x00;
    CCAP0L = 0xc0;  //Set the initial value same as CCAP0H
    CCAP0H = 0xc0;  //25% Duty Cycle
    CCAPM0 = 0x42;  //0100,0010  Setup PCA module 0 in PWM mode
    CR = 1; //Start PCA Timer.
    while(1){};
 }
```

8.6 PCA/PWM新增特殊功能寄存器声明

```
;STC12C5201AD 特殊功能寄存器头文件，STC12C5201_PCA_SFR.ASM
;声明 STC12C5201AD 系列 MCU 特殊功能寄存器地址
IPH      EQU   0B7H           ;中断优先级高位寄存器
CH       EQU   0F9H           ;PCA 计数器高 8 位。
CL       EQU   0E9H           ;PCA 计数器低 8 位。
;-----------------------
CCON     EQU   0D8H           ;PCA 控制寄存器。
CCF0     EQU   CCON.0         ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1     EQU   CCON.1         ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2     EQU   CCON.2         ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3     EQU   CCON.3         ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4     EQU   CCON.4         ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5     EQU   CCON.5         ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。
CR       EQU   CCON.6         ;1:允许 PCA 计数器计数，必须由软件清 0。
CF       EQU   CCON.7         ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                              ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
;-----------------------
CMOD     EQU   0D9H           ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1   CPS0
;            0      0     外部晶体频率 /12。
;            0      1     外部晶体频率 /2。
;            1      0     Timer 0 溢出脉冲，
;                         Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;            1      1     从 ECI/P3.4 脚输入的外部时钟(STC12C5A60S2: ECI/P1.2)。

;CMOD.0     ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;-----------------------
CCAP0H   EQU   0FAH           ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H   EQU   0FBH           ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H   EQU   0FCH           ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H   EQU   0FDH           ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H   EQU   0FEH           ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H   EQU   0FFH           ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L   EQU   0EAH           ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L   EQU   0EBH           ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L   EQU   0ECH           ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L   EQU   0EDH           ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L   EQU   0EEH           ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L   EQU   0EFH           ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。
```

```
;-----------------------
PCA_PWM0 EQU   0F2H            ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU   0F3H            ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU   0F4H            ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU   0F5H            ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU   0F6H            ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU   0F7H            ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:   7     6     5     4     3     2     1      0
;            -     -     -     -     -     -   EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;-----------------------
CCAPM0   EQU   0DAH            ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH            ;PCA 模块 1 的工作模式寄存器。
CCAPM2   EQU   0DCH            ;PCA 模块 2 的工作模式寄存器。
CCAPM3   EQU   0DDH            ;PCA 模块 3 的工作模式寄存器。
CCAPM4   EQU   0DEH            ;PCA 模块 4 的工作模式寄存器。
CCAPM5   EQU   0DFH            ;PCA 模块 5 的工作模式寄存器。

;CCAPMn:   7     6      5      4     3     2     1     0
;          -   ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0     0     0     0     0    00H 未启用任何功能。
;  x      1      0     0     0     0     x    21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1     0     0     0     x    11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1     0     0     0     x    31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0     1     0     0     x    49H 16 位软件定时器。
;  1      0      0     1     1     0     x    4DH 16 位高速脉冲输出。
;  1      0      0     0     0     1     0    42H 8 位 PWM。
;------------------------------------------------------------------------------
```

8.7 PWM 输出汇编语言示例程序

```
;/* --- STC International Limited ------------------------------------------ */
;/* --- 宏晶科技     姚永平     2006/1/6     V1.0 ------------------------------ */
;/* --- 使用 STC12C5201AD 系列单片机 PWM 输出汇编语言示例程序 ---------------- */
;/* --- Mobile: 13922805190 ------------------------------------------------ */
;/* --- Fax: 0755-82944243 ------------------------------------------------- */
;/* --- Tel: 0755-82948409 ------------------------------------------------- */
;/* --- Web: www.mcu-memory.com -------------------------------------------- */
;/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ------------ */
;/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ---- */
;/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 ---- */


;STC12C5201AD 系列单片机 PCA 功能  PWM 示例程序，使用 18.432MHz 晶振。
;-------------------------------------------------------------
#include <..\STC12_PCA_SFR.ASM>    ;定义 PCA 特殊功能寄存器
;-------------------------------------------------------------

;定义常量
;pulse_width_MAX = pulse_width_MIN 时，输出脉冲宽度不变。
pulse_width_MAX   EQU   0F0H        ;PWM 脉宽最大值，占空比 = 93.75%
pulse_width_MIN   EQU   10H         ;PWM 脉宽最小值，占空比 = 6.25%
step              EQU   38H         ;PWM 脉宽变化步长
;-------------------------------------------------------------
;定义变量
pulse_width     EQU   30H
;-------------------------------------------------------------
    ORG   0000H
    AJMP  main
;-------------------------------------------------------------
    ORG   0050H
main:
    MOV   SP, #0E0H

    ACALL PCA_init
main_loop:
    ACALL PWM
    SJMP  main_loop
;-------------------------------------------------------------
```

```
PCA_init:
    MOV    CMOD, #80H;             ;PCA 在空闲模式下停止 PCA 计数器工作
                                   ;PCA 时钟模式为 fosc/12
                                   ;禁止 PCA 计数器溢出中断
    MOV    CCON, #00H              ;禁止 PCA 计数器工作，清除中断标志、计数器溢出标志
    MOV    CL, #00H                ;清 0 计数器
    MOV    CH, #00H
;------------------------
;设置模块 0 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.7(第 11 脚)输出
    MOV    CCAPM0, #42H            ;*** 示例程序核心语句，---->0100,0010
    MOV    PCA_PWM0, #00H          ;*** 示例程序核心语句
;   MOV    PCA_PWM0, #03H          ;释放本行注释，PWM 输出就一直是 0，无脉冲。

;------------------------
;设置模块 1 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.5(第 9 脚)输出
    MOV    CCAPM1, #42H            ;*** 示例程序核心语句，---->0100,0010
    MOV    PCA_PWM1, #00H          ;*** 示例程序核心语句
;   MOV    PCA_PWM1, #03H          ;释放本行注释，PWM 输出就一直是 0，无脉冲。

;    SETB   EPCA_LVD               ;开 PCA 中断
;    SETB   EA                     ;开总中断
    SETB   CR                      ;将 PCA 计数器打开
    RET

;------------------------------------------------------------
PWM:                               ;用示波器进行观察较为理想。

     ;逐渐变亮。
    MOV    A, #pulse_width_MIN     ;为输出脉冲宽度设置初值。
    MOV    pulse_width, A          ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop1:
    MOV    A, pulse_width          ;判是否到达最大值。
    CLR    C
    SUBB   A, #pulse_width_MAX
    JNC    PWM_a                   ;到达最大值就转到逐渐变暗。
    MOV    A, pulse_width          ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
    MOV    CCAP0H, A               ;*** 示例程序核心语句
    MOV    CCAP1H, A               ;*** 示例程序核心语句

    CPL    A                       ;用 P1 口的 LED 显示占空比，
    MOV    P1, A                   ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV    A, pulse_width          ;计算下一次输出脉冲宽度数值。
    ADD    A, #step
    MOV    pulse_width, A
    ACALL  delay                   ;在一段时间内保持输出脉冲宽度不变。
    SJMP   PWM_loop1
```

```
PWM_a:
     ;逐渐变暗。
    MOV   A, #pulse_width_MAX   ;为输出脉冲宽度设置初值。
    MOV   pulse_width, A        ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop2:
    MOV   A, pulse_width        ;判是否到达最小值。
    CLR   C
    SUBB  A, #pulse_width_MIN
    JC    PWM_b                 ;到达最小值就返回。
    JZ    PWM_b                 ;到达最小值就返回。
    MOV   A, pulse_width        ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
    MOV   CCAP0H, A             ;*** 示例程序核心语句
    MOV   CCAP1H, A             ;*** 示例程序核心语句

    CPL   A                     ;用 P1 口的 LED 显示占空比,
    MOV   P1, A             ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV   A, pulse_width    ;计算下一次输出脉冲宽度数值。
    CLR   C
    SUBB  A, #step
    MOV   pulse_width, A
    ACALL delay             ;在一段时间内保持输出脉冲宽度不变。
    SJMP  PWM_loop2
PWM_b:
    RET
;-------------------------------------------------------------

delay:
    CLR   A
    MOV   R1, A
    MOV   R2, A
    MOV   R3, #80H
delay_loop:
    NOP
    NOP
    NOP
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    DJNZ  R3, delay_loop
    RET
;-------------------------------------------------------------
    END
```

8.8 用PCA做高速脉冲输出示例程序

```
;/* --- STC International Limited ------------------------------------------- */
;/* --- 宏晶科技    姚永平    2006/1/6    V1.0 ------------------------------ */
;/* --- 使用 STC12C5201AD 系列单片机 高速脉冲输出功能汇编语言示例程序 -------- */
;/* --- Mobile: 13922805190 ---------------------------------------------------- */
;/* --- Fax: 0755-82944243 ----------------------------------------------------- */
;/* --- Tel: 0755-82948409 ----------------------------------------------------- */
;/* --- Web: www.mcu-memory.com ---------------------------------------------- */
;/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ------------ */
;/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ---- */
;/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 ---- */
;*******************************************************************************
;                输出 125.0KHz 的脉冲(晶体频率 = 33.000MHz)
;
;示例程序: 使用 功能, 在 P3.5(第 9 脚)输出
;          125.0KHz 的方脉冲。
;-------------------------------------------------------------------------------
;    程序中定义的常量 CCAPnL_Value 决定了 PCA 模块 n 输出脉冲的频率 f：
;        f =  Fosc / (4 * CCAPnL_Value )
;           式中 Fosc = 晶体频率
;        CCAPnL_Value  =  Fosc / (4 * f)
;
;    如算出的结果不是整数, 则进行取整 CCAPnL_Value =  INT(Fosc / (4 * f) + 0.5)
;         INT() 为取整数运算, 直接去掉小数。
;*******************************************************************************
;定义 STC12C5201 系列 MCU 特殊功能寄存器
IPH       EQU    0B7H               ;中断优先级高位寄存器
CH        EQU    0xF9               ;PCA 计数器高 8 位。
CL        EQU    0xE9               ;PCA 计数器低 8 位。

;-----------------------
CCON      EQU    0D8H               ;PCA 控制寄存器。
CCF0      EQU    CCON.0             ;PCA 模块 0 中断标志, 由硬件置位, 必须由软件清 0。
CCF1      EQU    CCON.1             ;PCA 模块 1 中断标志, 由硬件置位, 必须由软件清 0。
CR        EQU    CCON.6             ;1:允许 PCA 计数器计数, 必须由软件清 0。
CF        EQU    CCON.7             ;PCA 计数器溢出标志,由硬件或软件置位,必须由软件清 0。

;-----------------------
```

```
CMOD      EQU    0D9H              ;PCA 工作模式寄存器。
;CMOD.7      CIDL: idle 状态时 PCA 计数器是否继续计数, 0: 继续计数, 1: 停止计数。

;CMOD.2      CPS1: PCA 计数器脉冲源选择位 1。
;CMOD.1      CPS0: PCA 计数器脉冲源选择位 0。
;            CPS1   CPS0
;             0      0     内部时钟, fosc/12。
;             0      1     内部时钟, fosc/2。
;             1      0     Timer0 溢出。
;             1      1     由 ECI/P3.4 脚输入的外部时钟。
;CMOD.0      ECF: PCA 计数器溢出中断允许位, 1-- 允许 CF(CCON.7) 产生中断。
;------------------------
CCAP0H    EQU    0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU    0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP0L    EQU    0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU    0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
;------------------------
PCA_PWM0 EQU    0F2H               ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU    0F3H               ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:     7      6      5      4      3      2      1      0
;              -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下, 与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下, 与 CCAPnL 组成 9 位数。
;------------------------
CCAPM0    EQU    0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1    EQU    0DBH              ;PCA 模块 1 的工作模式寄存器。

;CCAPMn:     7      6      5      4      3      2      1      0
;            -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时, 允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时, CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn CAPPn CAPNn MATn TOGn PWMn ECCFn
;  0     0     0     0    0    0    0    0x00   未启用任何功能。
;  x     1     0     0    0    0    x    0x21   16 位 CEXn 上升沿触发捕捉功能。
;  x     0     1     0    0    0    x    0x11   16 位 CEXn 下降沿触发捕捉功能。
;  x     1     1     0    0    0    x    0x31   16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1     0     0     1    0    0    x    0x49   16 位软件定时器。
;  1     0     0     1    1    0    x    0x4d   16 位高速脉冲输出。
;  1     0     0     0    0    1    0    0x42   8 位 PWM。
```

```
;------------------------------------------------------------
;定义常量 CCAPnL_Value
;CCAPnL_Value 决定了模块 1 输出脉冲的频率 f：
;        f =  Fosc / (4 * CCAPnL_Value )
;     式中 Fosc = 晶体频率
;     或  CCAPnL_Value  =  INT(Fosc / (4 * f) + 0.5)
;          INT() 为取整数运算。
;
;     假定 fosc = 20MHz 时, 要求 PCA 高速脉冲输出 125KHz 的方波:
;         CCAPnL_Value = INT( 20000000/4/125000 + 0.5)
;                      = INT( 40 + 0.5)
;                      = INT( 40.5 )
;                      = 40
;                      = 28H
;     输出脉冲的频率 f  = 20000000/4/40
;                      = 125000 (125.0KHz)

;CCAPnL_Value  EQU 25H     ;25H = 37, fosc = 18.432MHz 时, 高速脉冲输出 = 124.540KHz
;CCAPnL_Value  EQU 28H     ;28H = 40, fosc = 20MHz 时, 高速脉冲输出 = 125KHz
CCAPnL_Value   EQU 42H     ;42H = 66, fosc = 33MHz 时, 高速脉冲输出 = 125KHz
;------------------------------------------------------------
    ORG 0000H
    AJMP  main
;------------------------------------------------------------
    ORG   003BH                ;interrupt 7
PCA_interrupt:
    PUSH  ACC                  ;4 Clock
    PUSH  PSW                  ;4 Clock

    CLR   CCF1                 ;1 Clock, 清 PCA 模块 1 中断标志

    MOV   A, #CCAPnL_Value ;2 Clock
    ADD   A, CCAP1L            ;3 Clock
    MOV   CCAP1L, A            ;3 Clock
    CLR   A                    ;1 Clock
    ADDC  A, CCAP1H            ;3 Clock
    MOV   CCAP1H, A            ;3 Clock

    POP   PSW                  ;3 Clock
    POP   ACC                  ;3 Clock
    RETI                       ;4 Clock

;此中断服务程序共用 34 Clock, 进入中断服务程序还要数个 Clock
;------------------------------------------------------------
```

```
    ORG     0060H
main:
    MOV     SP, #0E0H               ;设置堆栈指针
    ACALL PCA_init                  ;调用 PCA 初始化程序

main_loop:
    NOP
    NOP
    NOP
    SJMP    main_loop
;-------------------------------------------------------------
PCA_init:                           ;PCA 初始化程序
    MOV     CMOD, #00000010B        ;02H, PCA 计数器在空闲模式下继续工作, CIDL = 0
                        ;PCA 计数器计数脉冲来源为系统时钟源 fosc/2, CPS1, CPS0 = (0,1)
                        ;禁止 PCA 计数器(CH, CL)计数溢出(CH, CL=0000H)中断, ECF = 0
    MOV     CCON, #00H              ;清除 PCA 计数器(CH, CL)计数溢出中断标志, CF = 0
                                    ;停止 PCA 计数器(CH, CL)计数, CR = 0
                                    ;清除 模块 1 中断标志, CCF1 = 0
                                    ;清除 模块 0 中断标志, CCF0 = 0
    MOV     CH, #00H                ;清 0 PCA 计数器高 8 位
    MOV     CL, #00H                ;清 0 PCA 计数器低 8 位
;-------------------------
;设置模块 1 为高速脉冲输出模式, 脉冲在 P3.5(第 9 脚)输出
    MOV     CCAPM1, #01001101B  ;4DH, 设置 PCA 模块 1 为高速脉冲输出模式,ECCF1=1,允许触发中断
;CCAPMn:    7       6       5       4       3       2       1       0
;           -     ECOMn   CAPPn   CAPNn   MATn    TOGn    PWMn    ECCFn
;           0       1       0       0       1       1       0       1

    MOV     CCAP1L, #CCAPnL_Value ;给模块 1 置初值, 此句不可少
    MOV     CCAP1H, #0 ;给模块 1 置初值, 此句不可少

    ;其它中断服务可能会使模块 1 高速脉冲输出的某个周期突然变得很大, 因此必须将
    ;PCA 中断的优先级设置为唯一的最高级, 其它中断的优先级都要比它低。
    MOV     IPH, #01000000B         ;PCA 中断的优先级设置为唯一的最高级
    MOV     IP, #01000000B

    SETB    EA                      ;开总中断
    SETB    CR                      ;将 PCA 计数器打开
    RET

;-------------------------------------------------------------
    END
;-------------------------------------------------------------
```

8.9　利用定时器 0 的溢出作为 PCA 模块的时钟输入源

--- 利用 PCA 模块 0 实现了可调频率的 PWM 输出

--- 利用 PCA 模块 1 重新实现了一个 16 位定时器

```
;/* ---  STC International Limited -------------------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 -------------------------------- */
;/* ---  使用 STC12C5201AD 系列单片机 定时器 0 的溢出，作为 PCA 模块的时钟输源 --  */
;/* ---  实现了可调频率的 PWM 输出,同时利用 PCA 模块再实现了定时器功能 ------------  */
;/* ---  Mobile: 13922805190 -------------------------------------------------- */
;/* ---  Fax: 0755-82944243 ---------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ---------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ----------------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 -------------  */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 -----  */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 -----  */
;-----------------------------------------------------------------------
;使用 定时器 0 的溢出，作为 PCA 模块的时钟输入源，利用 PCA 模块的多种功能
;实现了可调频率的 PWM 输出(还可以改变占空比)，同时利用 PCA 模块再实现了定时器功能
;使用 STC12C5201AD 系列单片机 PCA 模块的模块 0 的 PWM 功能 做 PWM 输出的示例程序
;使用 STC12C5201AD 系列单片机 PCA 模块的模块 1 的 16 位软定时器功能做定时器的示例程序
;晶振频率 Fosc = 18.432MHz，在 P1.5 输出脉冲宽度为 1 秒钟的方波
;--------------------------------------------------------------------------------
;声明 STC12C5201AD 系列 MCU 特殊功能寄存器地址
IPH         EQU     0B7H                ;中断优先级高位寄存器

CH          EQU     0F9H                ;PCA 计数器高 8 位。
CL          EQU     0E9H                ;PCA 计数器低 8 位。

;------------------------
CCON        EQU     0D8H                ;PCA 控制寄存器。
CCF0        EQU     CCON.0              ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1        EQU     CCON.1              ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2        EQU     CCON.2              ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3        EQU     CCON.3              ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4        EQU     CCON.4              ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5        EQU     CCON.5              ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。

CR          EQU     CCON.6              ;1:允许 PCA 计数器计数，必须由软件清 0。
CF          EQU     CCON.7              ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                                        ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
```

```
;-----------------------
CMOD      EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数, 0: 继续计数, 1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1   CPS0
;             0      0     外部晶体频率 /12。
;             0      1     外部晶体频率 /2。
;             1      0     Timer 0 溢出脉冲,
;                          Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;             1      1     从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0     ECF: PCA 计数器溢出中断允许位, 1-- 允许 CF(CCON.7) 产生中断。
;-----------------------

CCAP0H    EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H    EQU   0FCH              ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H    EQU   0FDH              ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H    EQU   0FEH              ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H    EQU   0FFH              ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L    EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L    EQU   0ECH              ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L    EQU   0EDH              ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L    EQU   0EEH              ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L    EQU   0EFH              ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。

;-----------------------
PCA_PWM0 EQU   0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU   0F3H              ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU   0F4H              ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU   0F5H              ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU   0F6H              ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU   0F7H              ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下, 与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下, 与 CCAPnL 组成 9 位数。
;-----------------------
```

```
CCAPM0   EQU   0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH              ;PCA 模块 1 的工作模式寄存器。
CCAPM2   EQU   0DCH              ;PCA 模块 2 的工作模式寄存器。
CCAPM3   EQU   0DDH              ;PCA 模块 3 的工作模式寄存器。
CCAPM4   EQU   0DEH              ;PCA 模块 4 的工作模式寄存器。
CCAPM5   EQU   0DFH              ;PCA 模块 5 的工作模式寄存器。
;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0      0     0     0     0    00H 未启用任何功能。
;  x      1      0      0     0     0     x    21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1      0     0     0     x    11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1      0     0     0     x    31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0      1     0     0     x    49H 16 位软件定时器。
;  1      0      0      1     1     0     x    4DH 16 位高速脉冲输出。
;  1      0      0      0     0     1     0    42H 8 位 PWM。
;-----------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_5mS_Flashing    EQU P1.6
LED_1S_Flashing     EQU P1.5
;-----------------------------------------------------------------------------
;定义常量
;Channe1_5mS_H, Channe1_5mS_L 的计算方法见 PCA 中断服务程序内的注释
;----------------------------------
;用定时器 0 的溢出率作 PCA 计数器(CH,CL)的时钟源时
;Channe1_5mS_H       EQU 03H     ;PCA 模块 1 5mS 定时常数高位, Fosc = 18.432
Channe1_5mS_H        EQU 01H     ;PCA 模块 1 5mS 定时常数高位, Fosc = 18.432
Channe1_5mS_L        EQU 00H     ;PCA 模块 1 5mS 定时常数低位, Fosc = 18.432
;Channe1_5mS_H       EQU 03H     ;PCA 模块 1 5mS 定时常数高位, Fosc = 22.1184
;Channe1_5mS_L       EQU 099H    ;PCA 模块 1 5mS 定时常数低位, Fosc = 22.1184
;-----------------
;内部时钟频率(fosc)/12 作 PCA 计数器(CH,CL)的时钟源
;Channe1_5mS_H       EQU 1EH     ;PCA 模块 1 5mS 定时常数高位
;Channe1_5mS_L       EQU 00H     ;PCA 模块 1 5mS 定时常数低位
;----------------------------------
Timer0_Reload_1      EQU 0F6H    ;Timer0 自动重装数 = -10
Timer0_Reload_2      EQU 0ECH    ;Timer0 自动重装数 = -20
```

```
;----------------------------------
PWM_PULSE_WIDTH     EQU 0FFH     ;数字越大脉宽越窄(占空比越小), P3.5 的 LED 越亮。
;--------------------------------------------------------------------------------
;定义变量
Counter             EQU 30H     ;声明一个计数器，用来计数中断的次数
;--------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;--------------------------------------------------------------------------------
    ORG   003BH                ;interrupt 7(0,1,2,3,4,5,6,7)
    LJMP  PCA_interrupt
;--------------------------------------------------------------------------------
    ORG   0050H
MAIN:
    CLR   LED_MCU_START        ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    MOV   Counter, #0          ;清 Counter 计数器
    ACALL PCA_Initiate         ;初始化 PCA
    ACALL Timer0_Initiate      ;初始化 T0
MAIN_Loop:
;#########  P3.5 的 LED 亮 #################
    MOV   TH0, #Timer0_Reload_1 ;T0 溢出率高
    MOV   TL0, #Timer0_Reload_1

    MOV   A, #PWM_PULSE_WIDTH   ;亮, 数字越大 PWM 占空比越小, P3.5 的 LED 越亮。
    MOV   CCAP0H, A
    ACALL delay
;--------------------
;请注意 T0 溢出率变低后定时器脉冲的 LED 闪烁速度变慢, 而 PWM 的 LED 亮度未改变
    MOV   TH0, #Timer0_Reload_2 ;T0 溢出率低
    MOV   TL0, #Timer0_Reload_2
    ACALL delay
;#########  P3.5 的 LED 较亮 ################
    MOV   TH0, #Timer0_Reload_1 ;T0 溢出率高
    MOV   TL0, #Timer0_Reload_1

    MOV   A, #PWM_PULSE_WIDTH
    ACALL RL_A                 ;改变 PWM 占空比
    ACALL RL_A
    MOV   CCAP0H, A             ;较亮, 数字越大 PWM 占空比越小, P3.5 的 LED 越亮
    ACALL delay
;--------------------
;请注意 T0 溢出率变低后定时器脉冲的 LED 闪烁速度变慢, 而 PWM 的 LED 亮度未改变
    MOV   TH0, #Timer0_Reload_2 ;T0 溢出率低
    MOV   TL0, #Timer0_Reload_2
    ACALL delay
    MOV   CCAP0H, A             ;暗, 数字越大 PWM 占空比越小, P3.5 的 LED 越亮
    ACALL delay
```

```
;######### P3.5 的 LED 暗 #################
    MOV   TH0, #Timer0_Reload_1 ;T0 溢出率高
    MOV   TL0, #Timer0_Reload_1

    MOV   A, #PWM_PULSE_WIDTH
    ACALL RL_A                  ;改变 PWM 占空比
    ACALL RL_A
    ACALL RL_A
    ACALL RL_A
;-------------------
;请注意 T0 溢出率变低后定时器脉冲的 LED 闪烁速度变慢，而 PWM 的 LED 亮度未改变
    MOV   TH0, #Timer0_Reload_2 ;T0 溢出率低
    MOV   TL0, #Timer0_Reload_2
    ACALL delay
;############################################
    SJMP  MAIN_Loop              ;无限循环。
;-----------------------------------------------------------------------------
RL_A:
    CLR   C
    RRC   A
    RET
;-----------------------------------------------------------------------------
Timer0_Initiate:
;初始化 T0，其溢出脉冲作 PCA 计数器(CH,CL)的时钟源
    MOV   TMOD, #02H             ;设置定时器 0 为自动重装工作模式
    MOV   TH0, #Timer0_Reload_1
    MOV   TL0, #Timer0_Reload_1
    SETB  TR0                    ;启动定时器 0
    RET
;-----------------------------------------------------------------------------
PCA_Initiate:
;   MOV   CMOD, #10000000B  ;PCA 在空闲模式下停止 PCA 计数器工作
;                           ;PCA 时钟源为 fosc/12
;                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CMOD, #10000100B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 定时器 0 (T0) 的溢出率
                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H        ;CF = 0，清 0 PCA 计数器溢出中断请求标志位
                           ;CR = 0，不允许 PCA 计数器计数
                            ;清 0 PCA 各模块中断请求标志位，如 CCF1，CCF0
    MOV   CL, #00H          ;清 0 PCA 计数器
    MOV   CH, #00H
;------------------------
;设置模块 0 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.7(第 11 脚)输出
    MOV   CCAPM0, #42H          ;*** 示例程序核心语句，设置模块 0 为 8 位 PWM 输出模式
    MOV   PCA_PWM0, #00H        ;*** 示例程序核心语句，清 0 PWM 模式下的第 9 位
;   MOV   PCA_PWM0, #03H        ;释放本行注释，PWM 输出就一直是 0，无脉冲。
```

```
;-------------------------
;设置 PCA 模块1
    ;Channe1_5mS_H, Channe1_5mS_L 的计算方法见 PCA 中断服务程序内的注释
    MOV    CCAP1L, #Channe1_5mS_L ;给 PCA 模块1 的 CCAP1L 置初值
    MOV    CCAP1H, #Channe1_5mS_H ;给 PCA 模块1 的 CCAP1H 置初值
    MOV    CCAPM1, #49H       ;设置 PCA 模块1 为16位软件定时器,ECCF1=1允许PCA模块1中断
     ;当[CH, CL]==[CCAP1H, CCAP1L]时，产生中断请求，CCF1=1，请求中断
    SETB   EA                 ;开整个单片机所有中断共享的总中断控制位
    SETB   CR                 ;启动 PCA 计数器(CH,CL)计数
     RET
;--------------------------------------------------------------------------------
PCA_Interrupt:
     PUSH   ACC
     PUSH   PSW

    CPL    LED_5mS_Flashing   ;本程序PCA模块1每5mS中断一次，每次进中断将该灯状态取反

;用定时器0 的溢出率作 PCA 计数器(CH,CL)的时钟源时，计算 Channe1_5mS_H, Channe1_5mS_L
    ;在本程序中定时器0 每12个时钟脉冲加1，定时器0每加 10 次后产生 1 次溢出，即每
    ;120 个时钟脉冲 PCA 计数器(CH,CL)加1。当[CH,CL] 增加到等于 [CCAP1H, CCAP1L]时
    ;CCF0=1, PCA 模块1产生中断请求。如果每次 PCA 模块1 中断后，在中断服务程序中给
     ;[CCAP1H, CCAP1L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe1_5mS_H, Channe1_5mS_L
    ;举例: 时钟频率 Fosc = 18.432MHz, PCA 计数器计数 300H 次等于 5mS。
    ;     Channe1_5mS_H, Channe1_5mS_L = T/( (1/Fosc)*120 )
    ;                                  = 0.005/ ( (1/18432000)*120 )
    ;                                  = 768 (10进制数)
    ;                                  = 300H (16进制数)
    ;     即 Channe1_5mS_H = 03H, Channe1_5mS_L = 00H
    ;
    ;    Channe1_5mS_H, Channe1_5mS_L：每次给 [CCAP1H,CCAP1L] 增加的数值(步长)

;内部时钟频率(fosc)/12 作 PCA 计数器(CH,CL)的时钟源,计算 Channe1_5mS_H,Channe1_5mS_L
    ;在本程序中[CH,CL]每12个时钟脉冲加1，当[CH,CL] 增加到等于 [CCAP1H, CCAP1L]时
    ;CCF0=1, PCA 模块1产生中断请求。如果每次 PCA 模块1 中断后，在中断服务程序中给
     ;[CCAP1H, CCAP1L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe1_5mS_H, Channe1_5mS_L
    ;举例: 时钟频率 Fosc = 18.432MHz, PCA 计数器计数 1E00H 次才是 5mS。
    ;     Channe1_5mS_H, Channe1_5mS_L = T/( (1/Fosc)*12 )
    ;                                  = 0.005/ ( (1/18432000)*12 )
    ;                                  = 7680 (10进制数)
    ;                                  = 1E00H (16进制数)
    ;     即 Channe1_5mS_H = 1EH, Channe1_5mS_L = 00H
     ;
    ;    Channe1_5mS_H, Channe1_5mS_L：每次给 [CCAP1H,CCAP1L] 增加的数值(步长)
```

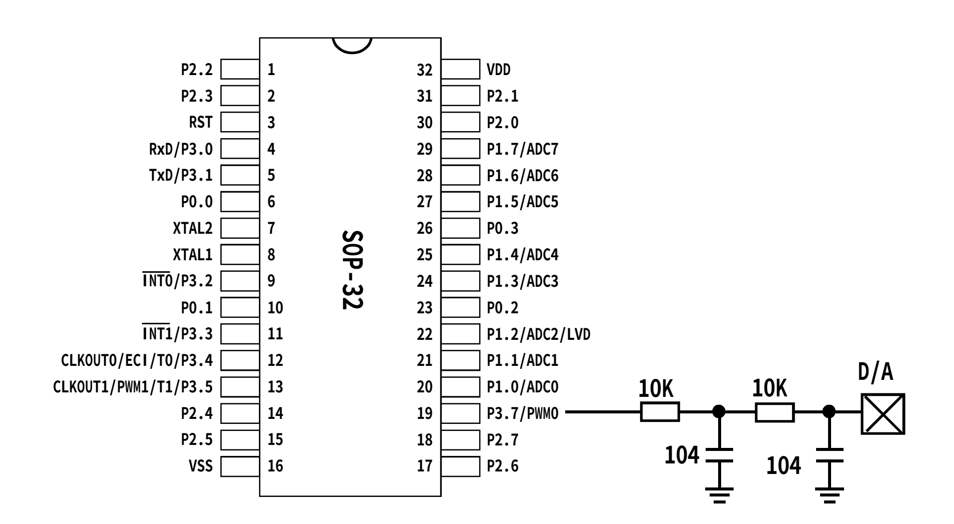
```
    MOV   A, #Channel_5mS_L      ;给[CCAP1H, CCAP1L] 增加一个数值
    ADD   A, CCAP1L
    MOV   CCAP1L, A
    MOV   A, #Channel_5mS_H
    ADDC  A, CCAP1H
    MOV   CCAP1H, A
    CLR   CCF1                   ;清 PCA 模块1 中断标志

    INC   Counter                ;中断次数计数器 + 1
    MOV   A, Counter
    CLR   C
    SUBB  A, #100                ;检测是否中断了 100 次 (0.5秒)
    JC    PCA_Interrupt_Exit     ;有借位,表示 Counter 小于 100,立即跳转退出

    MOV   Counter, #0            ;已中断了 100 次,清 0 中断次数计数器
    CPL   LED_1S_Flashing        ;在LED_1S_Flashing输出脉冲宽度为 0.5 秒钟的方波

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;-----------------------------------------------------------------------------
delay:
    CLR   A
    MOV   R1, A
    MOV   R2, A
    MOV   R3, #80H
delay_loop:
    NOP
    NOP
    NOP
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    DJNZ  R3, delay_loop
    RET
;-----------------------------------------------------------------------------
    END
```

8.10 利用PWM实现D/A功能的典型应用电路图

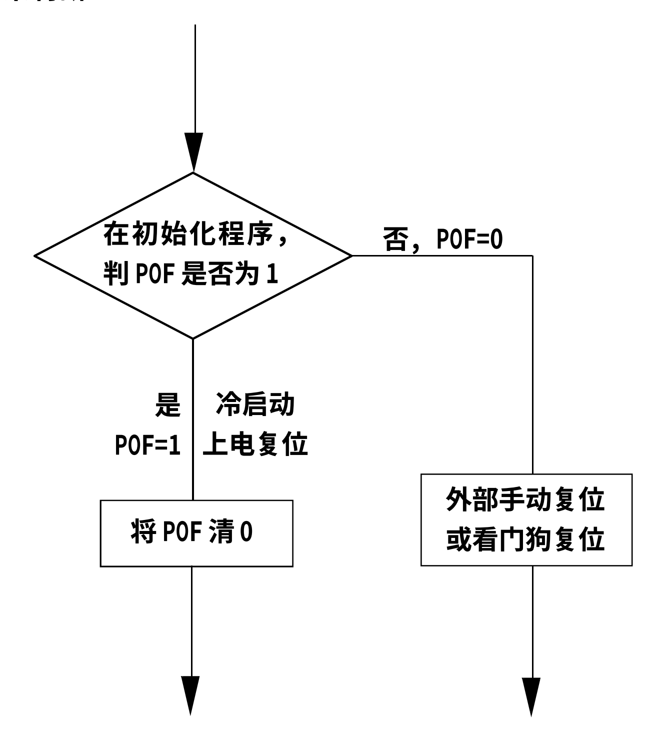


第九章 STC12系列单片机的掉电模式

9.1 PCON 寄存器的高级应用，上电复位标志，进入掉电模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

POF：上电复位标志位，单片机停电后，上电复位标志位为 1，可由软件清 0。
实际应用：要判断是上电复位（冷启动），还是外部复位脚输入复位信号产生的复位，还是内部看门狗复位，可通过如下方法来判断：



PD： 将其置 1 时，进入 Power Down 模式，可由外部中断低电平触发或下降沿触发中断模式唤醒,也可启动掉电唤醒专用定时器唤醒。
进入掉电模式时，外部时钟停振，CPU、定时器、串行口全部停止工作，只有外部中断继续工作。
可将 CPU 从掉电模式唤醒的外部管脚有：INT0/P3.2，INT1/P3.3，INT/T0/P3.4，INT/T1/P3.5，INT/RxD/P3.0

IDL：将其置 1，进入 IDLE 模式（空闲），除 CPU 不工作外，其余仍继续工作，可由任何一个中断唤醒。
可将 CPU 从空闲模式(IDLE 模式)唤醒的外部中断脚有：
INT0/P3.2，INT1/P3.3，INT/T0/P3.4,INT/T1/P3.5,INT/RxD/P3.0
内部定时器 Timer0,Timer1 也可以将单片机从空闲模式唤醒
串行口中断(UART)也可以将单片机从空闲模式唤醒

GF1,GF0: 两个通用工作标志位,用户可以任意使用。
SMOD： 波特率倍速位，置 1，串口通讯波特率快一倍

9.2 利用外部中断实现单片机从掉电模式唤醒(C语言)

```
/* --- STC International Limited ------------------------------------------- */
/* --- 宏晶科技    姚永平    2006/8/2    V1.0 -------------------------------- */
/* --- STC12xx 系列单片机,掉电模式唤醒测试程序(从外部中断 0 唤醒)--------------- */
/* --- Mobile: 13922805190 ------------------------------------------------ */
/* --- Fax: 0755-82944243 ------------------------------------------------- */
/* --- Tel: 0755-82948409 ------------------------------------------------- */
/* --- Web: www.mcu-memory.com -------------------------------------------- */
/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 --------------- */
/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 -------- */
/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --------- */

#include<reg52.h>
#include<intrins.h>

sbit  Begin_Led = P1^2;           // 系统开始工作指示灯
unsigned char Is_Power_Down = 0; // 进入 Power Down 之前，将其置为 1，以供判断
sbit  Is_Power_Down_Led_INT0     = P1^7; // 掉电唤醒指示灯，在外部中断 0 中
sbit  Not_Power_Down_Led_INT0    = P1^6; // 不是掉电唤醒指示灯，在外部中断 0 中
sbit  Is_Power_Down_Led_INT1     = P1^5; // 掉电唤醒指示灯，在外部中断 1 中
sbit  Not_Power_Down_Led_INT1    = P1^4; // 不是掉电唤醒指示灯，在外部中断 1 中
sbit  Power_Down_Wakeup_Pin_INT0 = P3^2; // 掉电唤醒管脚，外部中断 0
sbit  Power_Down_Wakeup_Pin_INT1 = P3^3; // 掉电唤醒管脚，外部中断 1
sbit  Normal_Work_Flashing_Led   = P1^3; // 系统处于正常工作状态指示灯


void Normal_Work_Flashing(void);
void INT_System_init(void);
void INT0_Routine(void);
void INT1_Routine(void);

void main(void)
{
    unsigned  char  j  =  0;
    unsigned  char  wakeup_counter  =  0;  // 中断唤醒次数变量初始为 0
    Begin_Led  =  0;       // 系统开始工作指示灯
    INT_System_init();     // 中断系统初始化
    while(1)
    {
        P2  = ~wakeup_counter;    // 中断唤醒次数显示,先将 wakeup_counter 取反
        wakeup_counter++;         // 中断唤醒次数显示
        for(j=0;j<2;j++)
        {
            Normal_Work_Flashing();   // 系统正常工作指示灯
        }
        Is_Power_Down  =  1; // 进入 Power Down 之前，将其置为 1，以供判断
        PCON  =  0x02; // 执行完此句，单片机进入 Power Down 模式，外部时钟停止振荡
```

```
        _nop_();
        //STC12 系列掉电模式，外部中断唤醒后，首先执行上句，然后才会进入中断服务程序
        _nop_();
        _nop_();  // 建议多加几个空操作指令 NOP
        _nop_();  // 建议多加几个空操作指令 NOP
    }
}

void INT_System_init(void)
{
    IT0   =   0;    /* 外部中断 0，低电平触发中断 */
//  IT0   =   1;    /* 外部中断 0，下降沿触发中断 */
    EX0   =   1;    /* 允许外部中断 0 中断 */
    IT1   =   0;    /* 外部中断 1，低电平触发中断 */
//  IT1   =   1;    /* 外部中断 1，下降沿触发中断 */
    EX1   =   1;    /* 允许外部中断 1 中断 */
    EA    =   1;    /* 开总中断控制位 */
}

void INT0_Routine(void) interrupt 0
{
    if(Is_Power_Down)
    {   //Is_Power_Down ==1,掉电唤醒,在外部中断 0 中
        Is_Power_Down  =   0;
        Is_Power_Down_Led_INT0  =  0;  // 点亮外部中断 0 掉电唤醒指示灯
        while(Power_Down_Wakeup_Pin_INT0==0)
        {
            /* 等待变高 */
        }
        Is_Power_Down_Led_INT0  =  1;  // 关闭外部中断 0 掉电唤醒指示灯
    }
    else
    {
        Not_Power_Down_Led_INT0 =  0;  // 点亮外部中断 0 正常工作中断指示灯
        while(Power_Down_Wakeup_Pin_INT0==0)
        {
            /* 等待变高 */
        }
        Not_Power_Down_Led_INT0 =  1;  // 关闭外部中断 0 正常工作中断指示灯
    }
}

void INT1_Routine(void) interrupt 2
{
    if(Is_Power_Down)
    {   //Is_Power_Down ==1,掉电唤醒,在外部中断 1 中
        Is_Power_Down   =   0;
        Is_Power_Down_Led_INT1 =  0;  // 顶亮外部中断 1 掉电唤醒指示灯
```

```
        while(Power_Down_Wakeup_Pin_INT1==0)
        {
            /* 等待变高 */
        }
        Is_Power_Down_Led_INT1  =  1;  // 关闭外部中断 1 掉电唤醒指示灯
    }
    else
    {
        Not_Power_Down_Led_INT1  =  0;  // 顶亮外部中断 1 正常工作中断指示灯
        while(Power_Down_Wakeup_Pin_INT1==0)
        {
            /* 等待变高 */
        }
        Not_Power_Down_Led_INT1  =  1;  // 关闭外部中断 1 正常工作中断指示灯
    }
}
void delay(void)
{
    unsigned int    j   =   0x00;
    unsigned int    k   =   0x00;
    for(k=0;k<2;++k)
    {
        for(j=0;j<=30000;++j)
        {
             _nop_();
             _nop_();
             _nop_();
             _nop_();
             _nop_();
             _nop_();
             _nop_();
             _nop_();
        }
    }
}
void Normal_Work_Flashing(void)
{
    Normal_Work_Flashing_Led    =   0;
    delay();
    Normal_Work_Flashing_Led    =   1;
    delay();
}
```

9.3 通过外部中断从掉电模式唤醒

```
;*************************************************************************
;Wake Up Idle and Wake Up Power Down
;*************************************************************************
    ORG     0000H
    AJMP    MAIN

    ORG     0003H
int0_interrupt:
    CLR     P1.7            ;点亮 P1.7 LED 表示已响应 int0 中断
    ACALL   delay           ;延时是为了便于观察，实际应用不需延时
    CLR     EA              ;关闭中断，简化实验. 实际应用不需关闭中断
    RETI

    ORG     0013H
int1_interrupt:
    CLR     P1.6            ;点亮 P1.6 LED 表示已响应 int1 中断
    ACALL   delay           ;延时是为了便于观察，实际应用不需延时
    CLR     EA              ;关闭中断，简化实验. 实际应用不需关闭中断
    RETI

    ORG     0100H
delay:
    CLR     A
    MOV     R0, A
    MOV     R1, A
    MOV     R2, #02
delay_loop:
    DJNZ    R0, delay_loop
    DJNZ    R1, delay_loop
    DJNZ    R2, delay_loop
    RET

main:
    MOV     R3, #0          ;P1 LED 递增方式变化，表示程序开始运行
main_loop:
    MOV     A, R3
    CPL     A
    MOV     P1, A
    ACALL   delay
```

```
    INC     R3
    MOV     A, R3
    SUBB    A, #18H
    JC      main_loop

    MOV     P1, #0FFH           ;熄灭全部灯表示进入 Power Down 状态

    CLR     IT0                 ;设置低电平激活外部中断
;   SETB    IT0

    SETB    EX0                 ;允许外部中断 0

    CLR     IT1                 ;设置低电平激活外部中断
;   SETB    IT1
    SETB    EX1                 ;允许外部中断 1

    SETB    EA                  ;开中断，若不开中断就不能唤醒 Power Down

;下条语句将使 MCU 进入 idle 状态或 Power Down 状态
;低电平激活外部中断可以将 MCU 从 Power Down 状态中唤醒
;其方法为:将外部中断脚拉低

    MOV     PCON, #00000010B        ;令 PD=1，进入 Power Down 状态，PD = PCON.1

    ;MOV     PCON, #00000001B       ;删除本语句前的 ";"，同时将前 1 条语句前加上注释符号 ";",
                                    ;令 IDL=1，可进入 idle 状态，IDL = PCON.0

    MOV     P1, #0DFH           ;1101,1111  请注意:
                                ; 1.外部中断使 MCU 退出 Power Down 状态,执行本条指令后
                                ;响应中断，表现为 P1.5 与 P1.7 的 LED 同时亮(INT0 唤醒)
                                ; 2.外部中断使 MCU 退出 idle 状态,先响应中断然后再执行本
                                ;条指令，表现为 P1.7 的 LED 先亮(INT0 唤醒)P1.5 的 LED 后亮
                                ; 3.实际使用掉电模式时,本语句应用 NOP 代替
    NOP         ;实际使用掉电模式时,应在 MOV    PCON, #00000010B 语句后面多加几个 NOP
    NOP         ;实际使用掉电模式时,应在 MOV    PCON, #00000010B 语句后面多加几个 NOP
    NOP         ;实际使用掉电模式时,应在 MOV    PCON, #00000010B 语句后面多加几个 NOP
WAIT1：
    SJMP    WAIT1               ;跳转到本语句，停机
    END
```

第十章 STC12C5201AD系列单片机电气特性

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

Parameter	Symbol	MIN	MAX	UNIT
Storage temperature	T_{ST}	-55	+125	℃
Operating Temperature(I)	T_A	-40	+85	℃
Operating Temperature(C)	T_A	0	+70	℃
DC Power Supply(5V MCU)	V_{DD} - V_{SS}	-0.3	+5.5	V
DC Power Supply(3V MCU)	V_{DD} - V_{SS}	-0.3	+3.6	V
Voltage on any Pin		-0.3	VCC + 0.3	V

DC Specification(5V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	3.3	5.0	5.5	V	
I_{PWDN}	Power Down Current		<0.1		uA	5V
I_{IDLE}	Idle Current		3.0		mA	5V
I_{CC}	Operating Current		4 mA	20	mA	5V
V_{IL1}	Input low voltage (P0,P1,P2,P3)			0.8	V	5V
V_{IH1}	Input High voltage (P0,P1,P2,P3)	2.0			V	5V
V_{IH2}	Input High voltage (RESET)	2.2			V	5V
I_{OL1}	Sinking Current for Output Low (P0,P1,P2,P3)		20		mA	5V Vpin=0.45V
I_{OH1}	(Quasi-output) Sourcing Current for Output high (P0,P1,P2,P3)	150	230		uA	5V
I_{OH2}	(Push-Pull,Strong-output) Sourcing Current for Output High (P0,P1,P2,P3)		20		mA	5V Vpin=2.4V
I_{IL}	Logic 0 input current (P0,P1,P2,P3)			50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P0,P1,P2,P3)	100	270	600	uA	V_{PIN}=2V

DC Specification(3.3V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	2.2	3.3	3.6	V	
I_{PWDN}	Power Down Current		<0.1		uA	3.3V
I_{IDLE}	Idle Current		2.0		mA	3.3V
I_{CC}	Operating Current		4 mA	10	mA	3.3V
V_{IL1}	Input low voltage (P0,P1,P2,P3)			0.8	V	3.3V
V_{IH1}	Input High voltage (P0,P1,P2,P3)	2.0			V	3.3V
V_{IH2}	Input High voltage (RESET)	2.2			V	3.3V
I_{OL1}	Sinking Current for Output Low (P0,P1,P2,P3)		20		mA	3.3V Vpin=0.45V
I_{OH1}	(QUasi-output) Sourcing Current for Output High (P0,P1,P2,P3)	40	70		uA	3.3V
I_{OH2}	(Push-Pull,Strong-output) Sourcing Current for Output High (P0,P1,P2,P3)		20		mA	3.3V
I_{IL}	Logic 0 input current (P0,P1,P2,P3)		8	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P0,P1,P2,P3)		110	600	uA	V_{PIN}=2V

第十一章 STC12 系列单片机开发 / 编程工具说明

11.1 在系统可编程（ISP）原理，官方演示工具使用说明

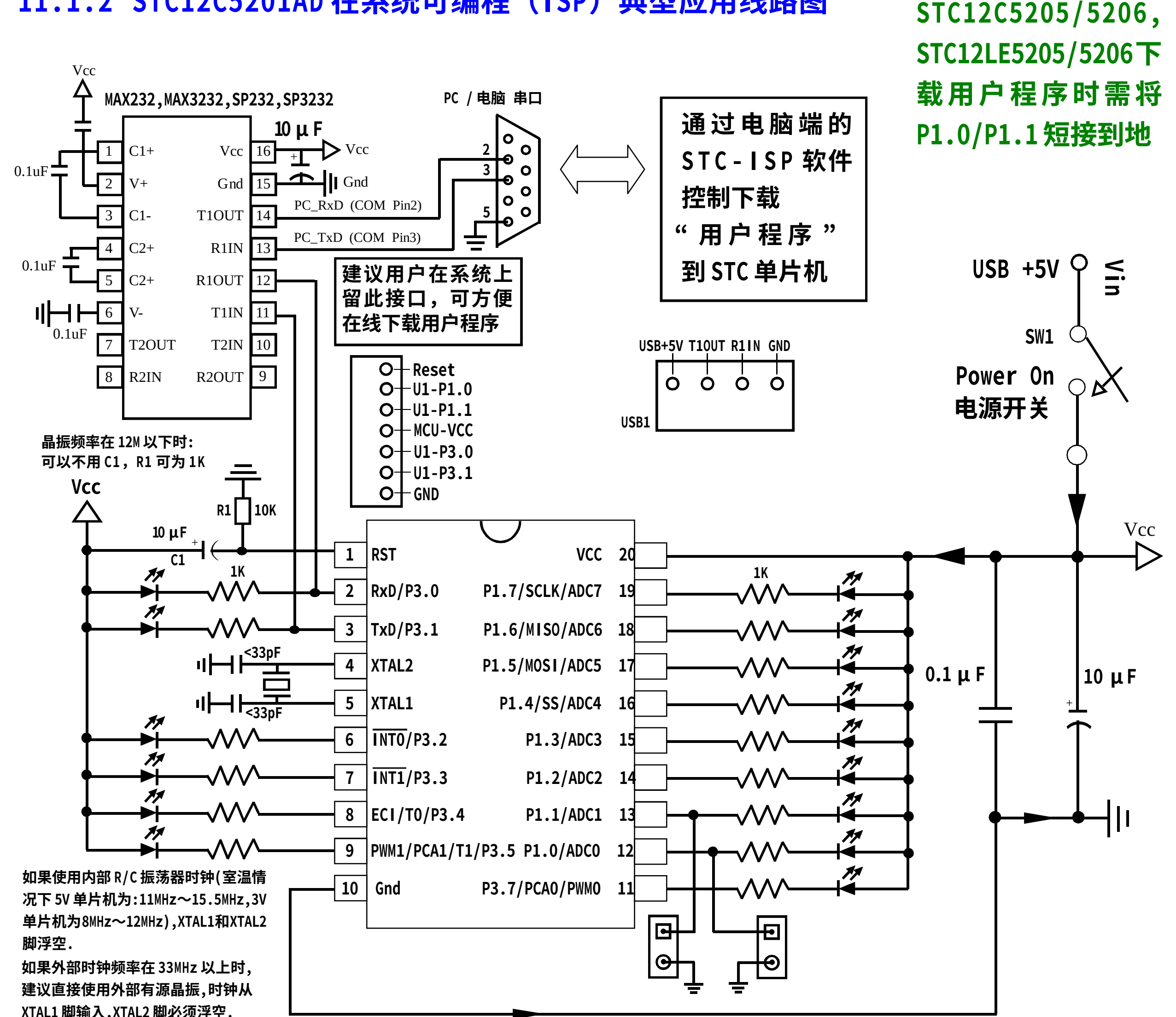
11.1.1 在系统可编程（ISP）原理使用说明

STC12C5205/5206, STC12LE5205/5206下载用户程序时需将P1.0/P1.1短接到地



11.1.2 STC12C5201AD 在系统可编程（ISP）典型应用线路图

STC12C5205/5206，STC12LE5205/5206下载用户程序时需将P1.0/P1.1短接到地



STC12C5201AD 系列单片机具有在系统可编程（ISP）特性，ISP 的好处是：省去购买通用编程器，单片机在用户系统上即可下载/烧录用户程序，而无须将单片机从已生产好的产品上拆下，再用通用编程器将程序代码烧录进单片机内部。有些程序尚未定型的产品可以一边生产，一边完善，加快了产品进入市场的速度，减小了新产品由于软件缺陷带来的风险。由于可以在用户的目标系统上将程序直接下载进单片机看运行结果对错，故无须仿真器。

STC12 系列单片机内部固化有 ISP 系统引导固件，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器(速度比通用编程器快，几秒一片)。

如何获得及使用 STC 提供的 ISP 下载工具（STC-ISP.exe 软件）：

(1). 获得 STC 提供的 ISP 下载工具（软件）

登陆 www.MCU-Memory.com 网站，从 STC 半导体专栏下载 PC（电脑）端的 ISP 程序，然后将其自解压，再安装即可（执行 setup.exe)，注意随时更新软件。

(2). 使用 STC-ISP 下载工具（软件)，请随时更新，目前已到 Ver3.1 版本以上，

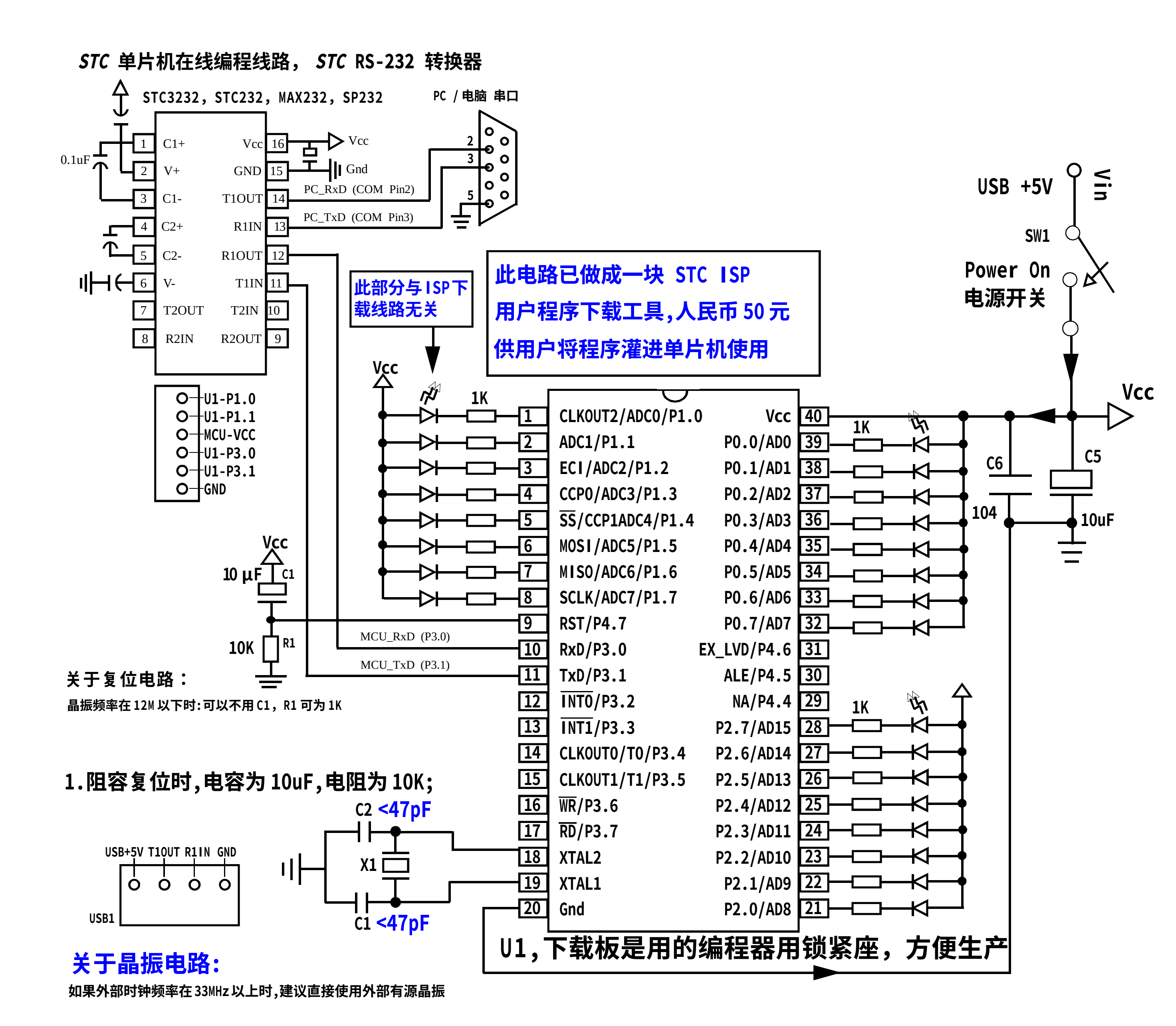
支持 *.bin,*.hex(Intel 16 进制格式)文件，少数 *.hex 文件不支持的话，请转换成 *.bin 文件

请随时注意升级 PC（电脑）端的 STC-ISP.EXE 程序。

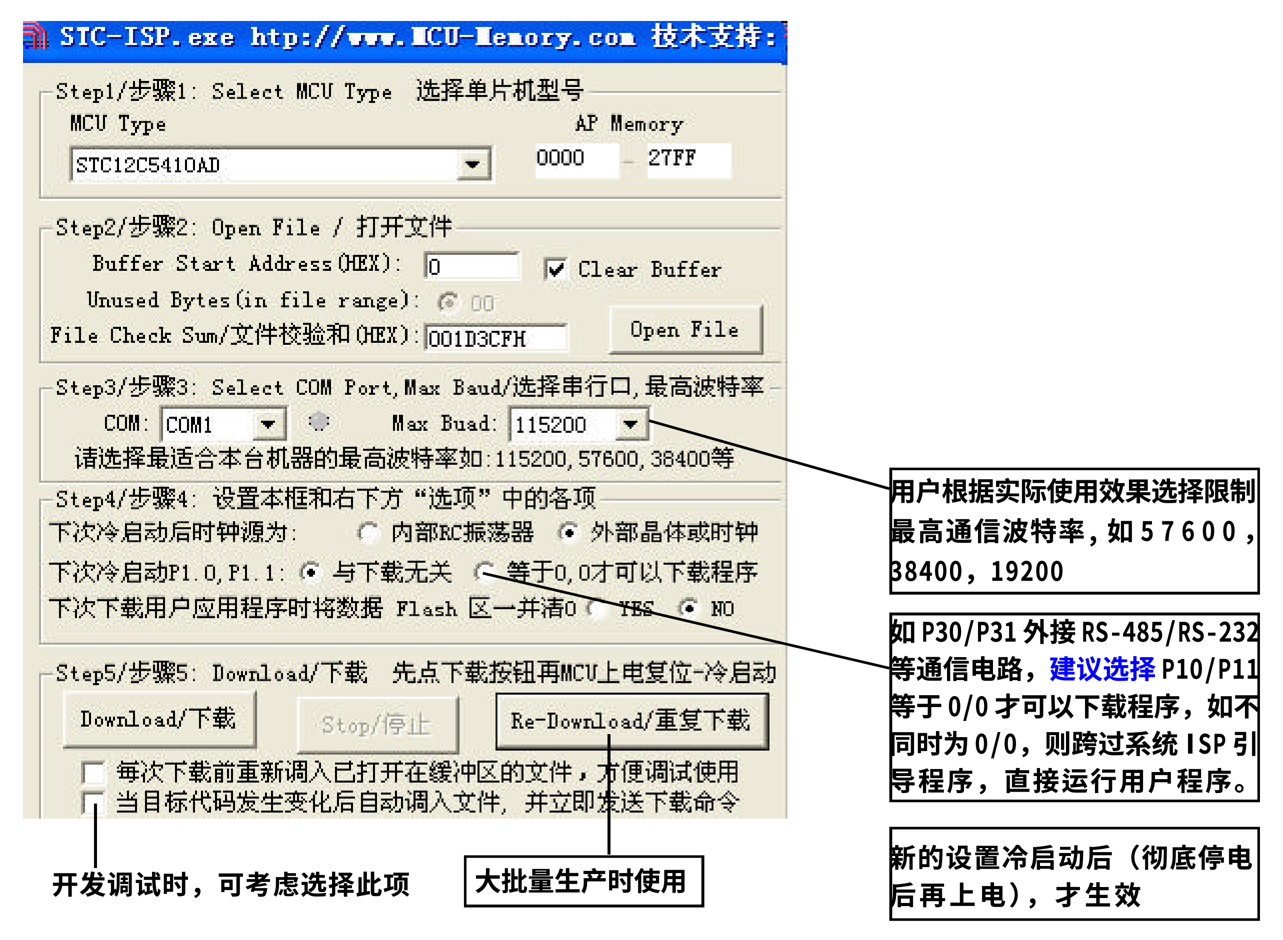
(3).STC12 系列单片机出厂时就已完全加密。需要单片机内部的电放光后上电复位(冷起动)才运行系统 ISP 程序，如从 P3.0/RxD 检测到合法的下载命令流就下载用户程序，如检测不到就复位到用户程序区，运行用户程序。

(4).如果用户板上 P3.0/RxD，P3.1/Txd 接了 RS-485 等电路，下载时需要将其断开。用户系统接了 RS-485 等通信电路，推荐在选项中选择“下次冷启动时需 P1.0/P1.1=0/0 才可以下载程序”

11.1.3 STC12C5A60S2 系列在系统可编程（ISP）典型应用线路图



11.1.4 电脑端的 ISP 控制软件界面使用说明



Step1/步骤1：选择你所使用的单片机型号，如STC12C5201等

Step2/步骤2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin， *.hex）

Step3/步骤3：选择串行口，你所使用的电脑串口，如串行口1--COM1， 串行口2--COM2,...

有些新式笔记本电脑没有RS-232串行口,可买一条USB-RS232转接器，人民币50元左右。

有些USB-RS232转接器，不能兼容，可让宏晶帮你购买经过测试的转换器。

Step4/步骤4：选择下次冷启动后，时钟源为"内部R/C振荡器"还是"外部晶体或时钟"。

Step5/步骤5：选择"Download/下载"按钮下载用户的程序进单片机内部，可重复执行

Step5/步骤5， 也可选择"Re-Download/重复下载"按钮

下载时注意看提示，主要看是否要给单片机上电或复位， 下载速度比一般通用编程器快。

一定要先选择"Download/下载"按钮，然后再给单片机上电复位(先彻底断电)，而不要先上电，先上电，检测不到合法的下载命令流，单片机就直接跑用户程序了。

关于硬件连接：

(1). MCU/单片机 RXD(P3.0) --- RS-232转换器 --- PC/电脑 TXD(COM Port Pin3)

(2). MCU/单片机 TXD(P3.1) --- RS-232转换器 --- PC/电脑 RXD(COM Port Pin2)

(3). MCU/单片机 GND ---------------------- PC/电脑 GND(COM Port Pin5)

(4). 如果您的系统P3.0/P3.1连接到 RS-485 电路，推荐

在选项里选择"下次冷启动需要P1.0/P1.1 = 0,0才可以下载用户程序"

这样冷启动后如 P1.0, P1.1 不同时为0,单片机直接运行用户程序，免得由于RS-485总线上的乱码造成单片机反复判断乱码是否为合法，浪费几百mS的时间，其实如果你的系统本身P3.0,P3.1就是做串口使用，也建议选择P1.0/P1.1 = 0/0才可下载用户程序，以便下次冷启动直接运行用户程序。

(5). RS-232转换器可选用MAX232/SP232(4.5-5.5V),MAX3232/SP3232(3V-5.5V).

11.1.5 宏晶科技的 ISP 下载编程工具硬件使用说明

如用户系统没有 RS-232 接口，
可使用STC-ISP Ver 3.0A.PCB演示板作为编程工具

STC-ISP Ver 3.0APCB 板可以焊接 3 种电路，分别支持 STC12 系列 16Pin / 20Pin / 28Pin / 32Pin。我们在下载板的反面贴了一张标签纸，说明它是支持 16Pin /20Pin / 28Pin / 32Pin 中的哪一种，用户要特别注意。在正面焊的编程烧录用锁紧座都是 40Pin 的，锁紧座第 20-Pin 接的是地线，请将单片机的地线对着锁紧座的地线插。

在 STC-ISP Ver 3.0A PCB 板完成下载编程用户程序的工作：

关于硬件连接：

(1). 根据单片机的工作电压选择单片机电源电压

A. 5V 单片机,短接 JP1 的 MCU-VCC, +5V 电源管脚

B. 3V 单片机,短接 JP1 的 MCU-VCC, 3.3V 电源管脚

(2).连接线(宏晶提供)

A. 将一端有 9 芯连接座的插头插入 PC/ 电脑 RS-232 串行接口插座用于通信

B. 将同一端的 USB 插头插入 PC/ 电脑 USB 接口用于取电

C. 将只有一个 USB 插头的一端插入宏晶的 STC-ISP Ver 3.0A PCB 板 USB1 插座用于 RS-232 通信和供电,此时 USB +5V Power 灯亮(D43,USB 接口有电)

(3).其他插座不需连接

(4).SW1 开关处于非按下状态，此时 MCU-VCC Power 灯不亮(D41)，没有给单片机通电

(5).SW3 开关

处于非按下状态，P1.0, P1.1 = 1, 1,不短接到地。

处于按下状态， P1.0, P1.1 = 0, 0,短接到地。

如果单片机已被设成“下次冷启动 P1.0/P1.1 = 0,0 才判 P3.0/RxD 有无合法下载命令流”

就必须将 SW3 开关处于按下状态，让单片机的 P1.0/P1.1 短接到地

(6).将单片机插进 U1-Socket 锁紧座，锁紧单片机,注意单片机是 20-Pin / 28-Pin,而 U1-Socket 锁紧座是 40-Pin,我们的设计是靠下插，靠近晶体的那一端插。

(7).关于软件：选择“Download/ 下载”(必须在给单片机上电之前让 PC 先发一串合法下载命令)

(8).按下 SW1 开关，给单片机上电复位，此时 MCU-VCC Power 灯亮(D41)

此时 STC 单片机进入 ISP 模式(STC12 系列冷启动进入 ISP)

(9).下载成功后，再按 SW1 开关，此时 SW1 开关处于非按下状态，MCU-VCC Power 灯不亮(D41)，给单片机断电，取下单片机，换上新的单片机。

11.1.6 用户板没有 RS-232 转换器,如何用宏晶科技的 ISP 下载板做 RS-232 通信转换

利用STC-ISP Ver 3.0A PCB 板进行RS-232转换
单片机在用户自己的板上完成下载 / 烧录：

1. U1-Socket 锁紧座不得插入单片机
2. 将用户系统上的电源(MCU-VCC,GND)及单片机的 P3.0/RXD,P3.1/TXD 接入转换板 CN2 插座
   这样用户系统上的单片机就具备了与 PC/ 电脑进行通信的能力
3. 将用户系统的单片机的 P1.0,P1.1 接入转换板 CN2 插座(如果需要的话)
4. 如须 P1.0, P1.1 = 0, 0,短接到地，可在用户系统上将其短接到地，或将 P1.0/P1.1 也从用户系统上引到 STC-ISP Ver3.0A PCB 板上，将 SW3 开关按下，则 P1.0/P1.1=0,0。
5. 关于软件：选择“Download/ 下载”
6. 给单片机系统上电复位(注意是从用户系统自供电，不要从电脑 USB 取电,电脑 USB 座不插)
7. 下载程序时，如用户板有外部看门狗电路，不得启动，单片机必须有正确的复位,但不能在 ISP 下载程序时被外部看门狗复位,如有，可将外部看门狗电路 WDI 端 / 或 WDO 端浮空
8. 如有 RS-485 晶片连到 P3.0/Rxd,P3.1/Txd,或其他线路，在下载时应将其断开。

11.2 编译器/汇编器，编程器，仿真器

STC 单片机应使用何种编译器/汇编器：

1.任何老的编译器/汇编器都可以支持，流行用Keil C51
2.把STC单片机，当成Intel的8052/87C52/87C54/87C58,Philips的P87C52/P87C54/P87C58就可以了
3.如果要用到扩展的专用特殊功能寄存器，直接对该地址单元设置就行了，当然先声明特殊功能寄存器的地址较好

编程烧录器:

我们有：STC12C5201AD/系列 ISP 经济型下载编程工具(人民币50元，可申请免费样品)
注意:有专门下载28PIN/20PIN的不同演示板,
28PIN是28PIN的演示板,20PIN是20PIN的演示板

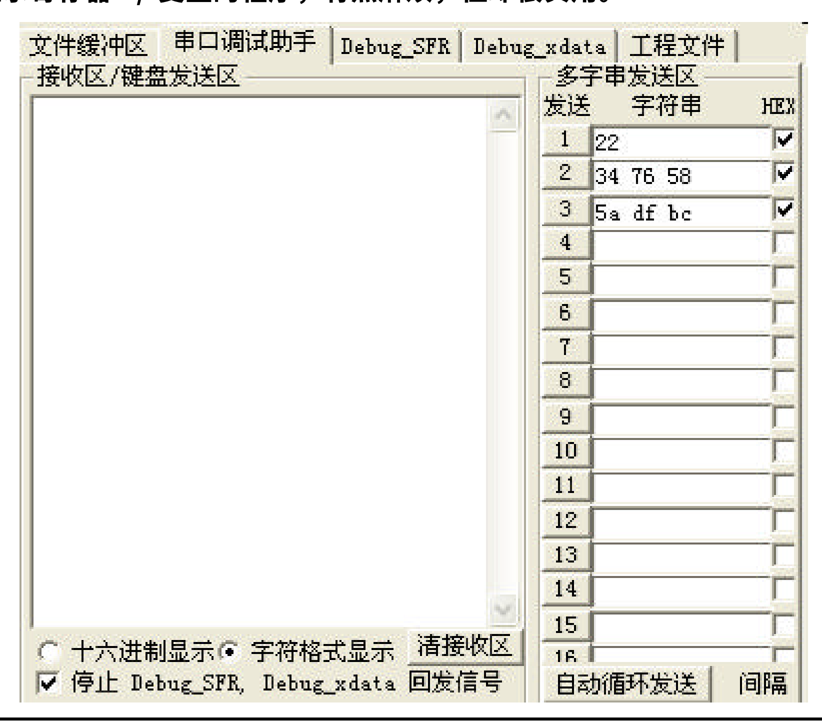
仿真器:如您已有老的仿真器，可仿真普通8052的基本功能

STC12C5201AD系列单片机扩展功能如它仿不了
可以用 STC-ISP.EXE 直接下载用户程序看运行结果就可以了,如需观察变量,可自己写一小段测试程序通过串口输出到电脑端的STC-ISP.EXE的“串口调试助手”来显示,也很方便。
无须添加新的设备

无仿真器如何调试/开发用户程序

1.首先参照本手册当中的“用定时器1做波特率发生器”，调通串口程序，这样，要观察变量就可以自己写一小段测试程序将变量通过串口输出到电脑端的STC-ISP.EXE的“串口调试助手”来显示,也很方便 。
2.调通按键扫描程序(到处都有大量的参考程序)
3.调通用户系统的显示电路程序，此时变量/寄存器也可以通过用户系统的显示电路显示了
4.调通A/D检测电路(我们用户手册里面有完整的参考程序)
5.调通PWM等电路(我们用户手册里面有完整的参考程序)
这样分步骤模块化调试用户程序，有些系统，熟练的8051用户，三天就可以调通了，难度不大的系统，一般一到二周就可以调通。

用户的串口输出显示程序可以在输出变量/寄存器的值之后，继续全速运行用户程序，也可以等待串口送来的“继续运行命令”，方可继续运行用户程序，这就相当于断点。这种断点每设置一个地方，就必须调用一次该显示寄存器 /变量的程序，有点麻烦，但却很实用。



11.3 自定义下载演示程序(实现不停电下载)

```
/* ---  STC International Limited ---------------------------------------------  */
/* ---  宏晶科技    姚永平    2006/7/31    V1.0 -------------------------------  */
/* ---  STC12C5201AD  系列单片机,软件实现自定义下载程序 ------------------------  */
/* ---  Mobile: 13922805190 ---------------------------------------------------  */
/* ---  Fax: 0755-82944243 ----------------------------------------------------  */
/* ---  Tel: 0755-82948409 ----------------------------------------------------  */
/* ---  Web: www.mcu-memory.com -----------------------------------------------  */
/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------------  */
/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 -------  */
/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 --------  */

#include<reg52.h>
#include<intrins.h>
sfr  IAP_CONTR =  0xC7;
sfr  CCON     = 0xD8;
sfr  CMOD     = 0xD9;
sfr  CL       = 0xE9;
sfr  CH       = 0xF9;
sfr  CCAP0L   = 0xEA;
sfr  CCAP0H   = 0xFA;
sfr  CCAPM0   = 0xDA;
sfr  CCAPM1   = 0xDB;
sbit CR       = 0xDE;
sbit MCU_Start_Led  =  P1^7;
//unsigned char self_command_array[4] =  {0x22,0x33,0x44,0x55};
#define Self_Define_ISP_Download_Command  0x22
#define RELOAD_COUNT  0xfb   //18.432MHz,12T,SMOD=0,9600bps

void serial_port_initial();
void send_UART(unsigned char);
void UART_Interrupt_Receive(void);
void soft_reset_to_ISP_Monitor(void);
void delay(void);
void display_MCU_Start_Led(void);
void send_PWM(void);

void main(void)
{
    unsigned  char i = 0;
    serial_port_initial();       //串口初始化
    display_MCU_Start_Led();     //点亮发光二极管表示单片机开始工作
    send_UART(0x34);             //串口发送数据表示单片机串口正常工作
    send_UART(0xa7);             //串口发送数据表示单片机串口正常工作
    send_PWM();                  //6kHz PWM,  50% duty
    while(1);
```

```
}

void serial_port_initial()
{
    SCON    =   0x50;    //0101,0000 8位可变波特率，无奇偶校验位
    TMOD    =   0x21;    //0011,0001 设置顶时器1为8位自动重装计数器
    TH1     =   RELOAD_COUNT;    //设置定时器1自动重装数
    TL1     =   RELOAD_COUNT;
    TR1     =   1;      //开定时器1
    ES      =   1;      //允许串口中断
    EA      =   1;      //开总中断
}

void send_UART(unsigned char i)
{
    ES      =   0;  //关串口中断
    TI      =   0;  //清零串口发送完成中断请求标志
    SBUF    =   i;
    while(TI ==0); //等待发送完成
    TI      =   0;  //清零串口发送完成中断请求标志
    ES      =   1;  //允许串口中断
}

void UART_Interrupt_Receive(void) interrupt 4
{
    unsigned char   k   =   0;
    if(RI==1)
    {
        RI  =   0;
        k   =   SBUF;
        if(k==Self_Define_ISP_Download_Command)    //是自定义下载命令
        {
            delay();     //延时1秒就足够了
            delay();     //延时1秒就足够了
            soft_reset_to_ISP_Monitor();     //软复位到系统ISP监控区
        }
        send_UART(k);
    }
    else
    {
        TI  =   0;
    }
}

void soft_reset_to_ISP_Monitor(void)
{
    IAP_CONTR   =    0x60;    //0110,0000    软复位到系统ISP监控区
```

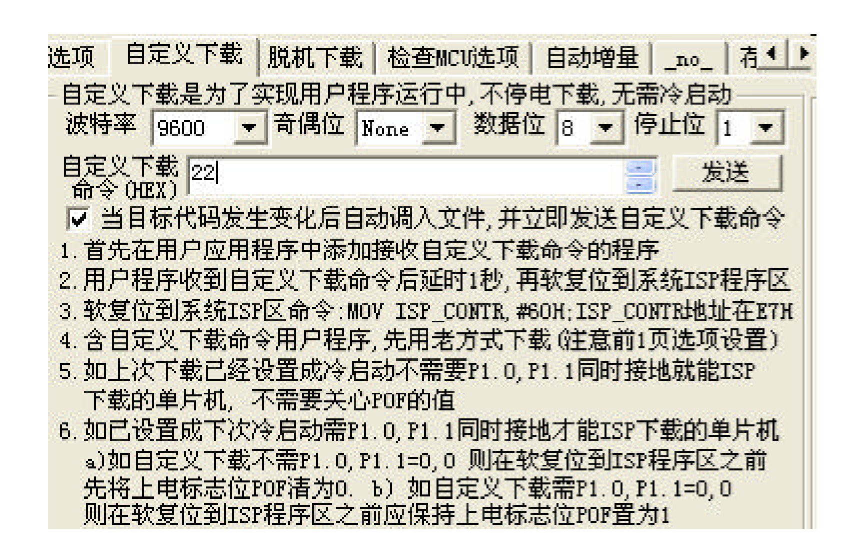
```
}

void delay(void)
{
    unsigned int j  =   0;
    unsigned int g  =   0;
    for(j=0;j<5;j++)
    {
        for(g=0;g<60000;g++)
        {
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
        }
    }
}

void display_MCU_Start_Led(void)
{
    unsigned  char i = 0;
    for(i=0;i<3;i++)
    {
        MCU_Start_Led   =   0;  //顶亮MCU开始工作指示灯
        delay();
        MCU_Start_Led   =   1;  //熄灭MCU开始工作指示灯
        delay();
        MCU_Start_Led   =   0;  //顶亮MCU开始工作指示灯
    }
}

void send_PWM(void)
{
    CMOD    =   0x00;   // CIDL - - - - CPS1 CPS0 ECF  Setup PCA Timer
                        // CPS1 CPS0 = 00, Fosc/12 is PCA/PWM clock
                        // 18432000/12/256 = 6000
    CL      =   0x00;
    CH      =   0x00;
    CCAP0L  =   0x80;   //Set the initial value same as CCAP0H
    CCAP0H  =   0x80;   //50% Duty Cycle
    CCAPM0  =   0x42;   //0100,0010 Setup PCA module 0 in 8BIT PWM, P3.7
    CR      =   1;      //启动 PCA/PWM 定时器
}
```

自定义下载在 STC 的电脑端 ISP 软件 STC-ISP.EXE 中，还应做相应设置，具体参考设置见下图:



详细的帮助上图也有具体的说明

附录 A: 内部常规 256 字节 RAM 间接寻址测试程序

```
;/* --- STC International Limited ------------------------------------------ */
;/* --- 宏晶科技    姚永平    2006/1/6    V1.0 ---------------------------- */
;/* --- STC12C5201AD 系列单片机 内部常规 RAM 间接寻址测试程序 ----------------- */
;/* --- Mobile: 13922805190 ----------------------------------------------- */
;/* --- Fax: 0755-82944243 ------------------------------------------------ */
;/* --- Tel: 0755-82948409 ------------------------------------------------ */
;/* --- Web: www.mcu-memory.com ------------------------------------------- */
;/* --- 本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ----------- */
;/* --- 如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序  ---- */
;/* --- 如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序  ---- */

TEST_CONST EQU   5AH
;TEST_RAM  EQU   03H
     ORG   0000H
     LJMP  INITIAL

     ORG   0050H
INITIAL:
     MOV   R0,   #253

     MOV   R1,   #3H
TEST_ALL_RAM:
     MOV   R2,   #0FFH
TEST_ONE_RAM:
     MOV   A,    R2
     MOV   @R1,  A
     CLR   A
     MOV   A,    @R1

     CJNE  A,    2H,   ERROR_DISPLAY
     DJNZ  R2,   TEST_ONE_RAM
     INC   R1
     DJNZ  R0,   TEST_ALL_RAM

OK_DISPLAY:
     MOV   P1,   #11111110B
Wait1:
     SJMP  Wait1

ERROR_DISPLAY:
     MOV   A,    R1
     MOV   P1,   A
Wait2:
     SJMP  Wait2
     END
```

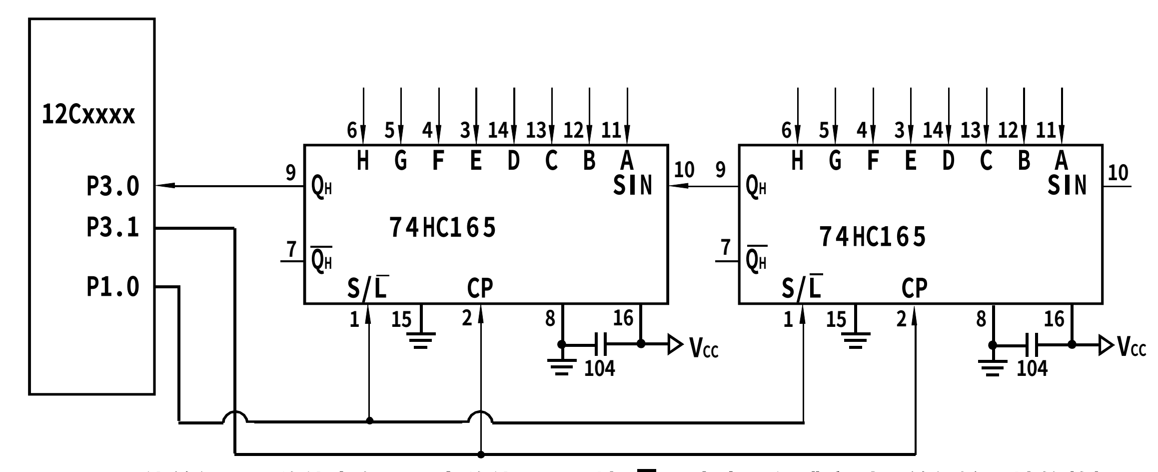
附录 B: 用串行口扩展 I/O 接口

STC12C5201 串行口的方式 0 可用于 I/O 扩展。如果在应用系统中，串行口未被占用，那么将它用来扩展并行 I/O 口是一种经济、实用的方法。

在操作方式 0 时，串行口作同步移位寄存器，其波特率是固定的，为 fosc/12（fosc 为振荡器频率)。数据由 RXD 端（P3.0）出入，同步移位时钟由 TXD 端（P3.1）输出。发送、接收的是 8 位数据，低位在先。

一、用 74HC165 扩展并行输入口

下图是利用两片 74HC165 扩展二个 8 位并行输入口的接口电路图。



74HC165 是 8 位并行置入移位寄存器。当移位 / 置入端($\overline{S}$/L)由高到低跳变时，并行输入端的数据置入寄存器；当 $\overline{S}$/L=1，且时钟禁止端（第 15 脚）为低电平时，允许时钟输入，这时在时钟脉冲的作用下，数据将由 Q_A 到 Q_H 方向移位。

上图中，TXD(P3.1)作为移位脉冲输出端与所有 74HC165 的移位脉冲输入端 CP 相连；RXD（P3.0）作为串行输入端与 74HC165 的串行输出端 Q_H 相连；P1.0 用来控制 74HC165 的移位与置入而同 S/$\overline{L}$相连；74HC165 的时钟禁止端（15 脚）接地，表示允许时钟输入。当扩展多个 8 位输入口时，两芯片的首尾（Q_H 与 S_{IN}）相连。

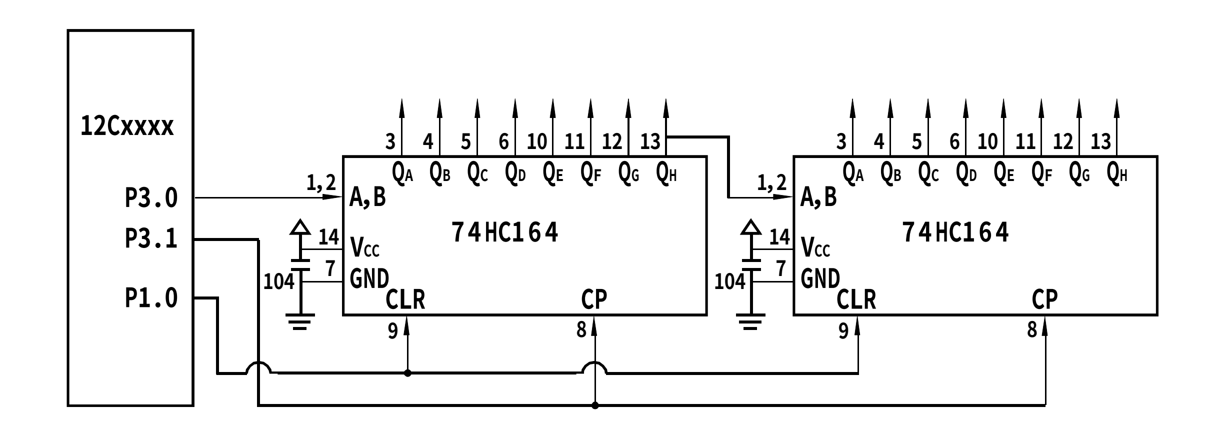
下面的程序是从 16 位扩展口读入 5 组数据（每组二个字节)，并把它们转存到内部 RAM 20H 开始的单元中。

```
         MOV   R7，#05H              ；设置读入组数
         MOV   R0，#20H              ；设置内部 RAM 数据区首址
START：  CLR   P1.0                  ；并行置入数据，S/L=0
         SETB  P1.0                  ；允许串行移位 S/L=1
         MOV   R1，#02H              ；设置每组字节数，即外扩 74LS165 的个数
RXDATA： MOV   SCON，#00010000B      ；设串行方式 0 ，允许接收，启动接收过程
WAIT：   JNB   RI，WAIT              ；未接收完一帧，循环等待
         CLR   RI                    ；清 RI 标志，准备下次接收
         MOV   A，SBUF               ；读入数据
         MOV   @R0，A                ；送至 RAM 缓冲区
         INC   R0                    ；指向下一个地址
         DJNZ  R1，RXDATA            ；为读完一组数据，继续
         DJNZ  R7，START             ；5 组数据未读完重新并行置入
         ……                          ；对数据进行处理
```

上面的程序对串行接收过程采用的是查询等待的控制方式，如有必要，也可改用中断方式。从理论上讲，按上图方法扩展的输入口几乎是无限的，但扩展的越多，口的操作速度也就越慢。

二、用 74HC164 扩展并行输出口

74HC164 是 8 位串入并出移位寄存器。下图是利用 74HC164 扩展二个 8 位输出口的接口电路。

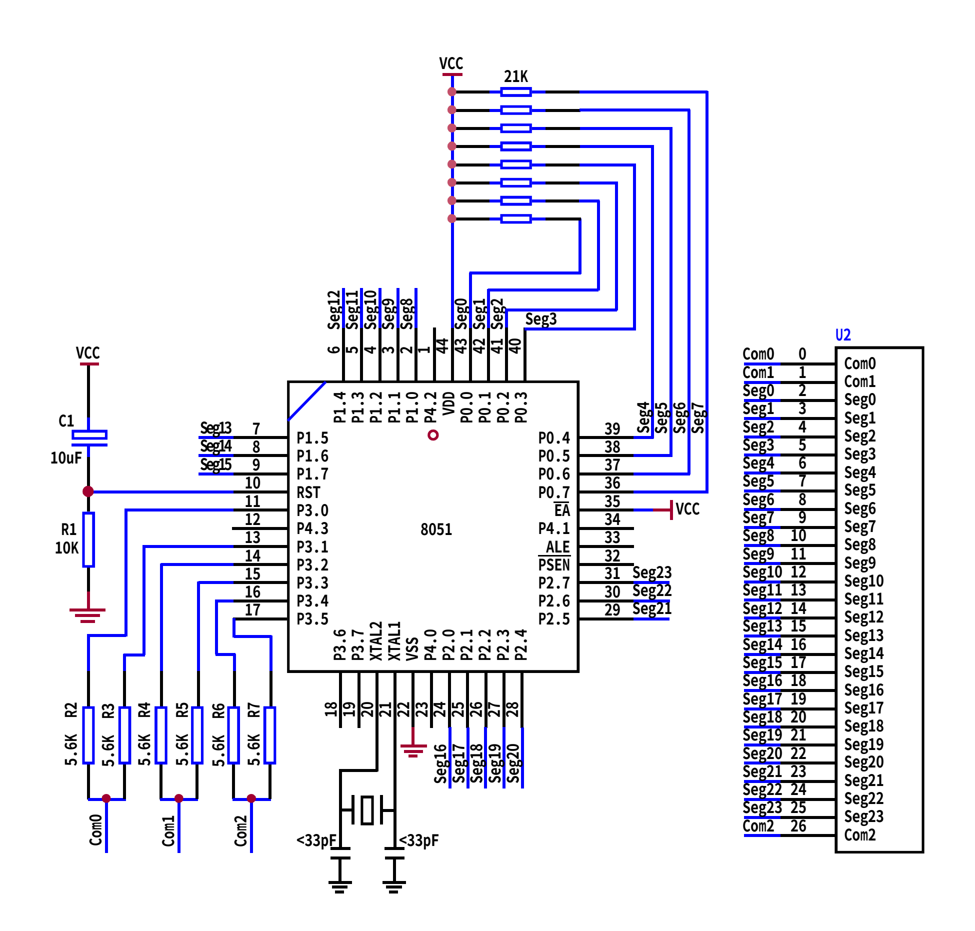


当单片机串行口工作在方式 0 的发送状态时，串行数据由 P3.0（RXD）送出，移位时钟由 P3.1（TXD）送出。在移位时钟的作用下，串行口发送缓冲器的数据一位一位地移入 74HC164 中。需要指出的是，由于 74HC164 无并行输出控制端，因而在串行输入过程中，其输出端的状态会不断变化，故在某些应用场合，在 74HC164 的输出端应加接输出三态门控制，以便保证串行输入结束后再输出数据。

下面是将 RAM 缓冲区 30H、31H 的内容串行口由 74HC164 并行输出的子程序。

```
START：  MOV    R7，#02H        ；设置要发送的字节个数
         MOV    R0，#30H        ；设置地址指针
         MOV    SCON，#00H      ；设置串行口方式 0
SEND：   MOV    A，@R0
         MOV    SBUF，A         ；启动串行口发送过程
WAIT：   JNB    TI，WAIT        ；一帧数据未发送完，循环等待
         CLR    TI
         INC    R0              ；取下一个数
         DJNZ   R7，SEND
         RET
```

附录C: 8051单片机普通I/O口驱动LCD显示



本资料不提供技术支持,请自行消化吸收

```
NAME LcdDriver
#include<reg52.h>

;**************************************************************************************
;the LCD is 1/3 duty and 1/3 bias; 3Com*24Seg; 9 display RAM;
;
;                   Bit7    Bit6   Bit5   Bit4   Bit3   Bit2   Bit1   Bit0
;Com0:  Com0Data0:  Seg7    Seg6   Seg5   Seg4   Seg3   Seg2  Seg1   Seg0
;       Com0Data1:  Seg15   Seg14  Seg13  Seg12  Seg11  Seg10        Seg9   Seg8
;       Com0Data2:  Seg23   Seg22  Seg21  Seg20  Seg19  Seg18        Seg17  Seg16
;Com1:  Com1Data0:  Seg7    Seg6   Seg5   Seg4   Seg3   Seg2  Seg1   Seg0
;       Com1Data1:  Seg15   Seg14  Seg13  Seg12  Seg11  Seg10        Seg9   Seg8
;       Com1Data2:  Seg23   Seg22  Seg21  Seg20  Seg19  Seg18        Seg17  Seg16
;Com2:  Com2Data0:  Seg7    Seg6   Seg5   Seg4   Seg3   Seg2  Seg1   Seg0
;       Com2Data1:  Seg15   Seg14  Seg13  Seg12  Seg11  Seg10        Seg9   Seg8
;       Com2Data2:  Seg23   Seg22  Seg21  Seg20  Seg19  Seg18        Seg17  Seg16
;**************************************************************************************
;Com0:  P3^0,P3^1   when P3^0 = P3^1 = 1      then Com0=VCC(=5V);
;                        P3^0 = P3^1 = 0      then Com0=GND(=0V);
;                        P3^0 = 1, P3^1=0     then Com0=1/2 VCC;
;Com1:  P3^2,P3^3   the same as the Com0
;Com2:  P3^4,P3^5   the same as the Com0
;

sbit SEG0  =P0^0
sbit SEG1  =P0^1
sbit SEG2  =P0^2
sbit SEG3  =P0^3
sbit SEG4  =P0^4
sbit SEG5  =P0^5
sbit SEG6  =P0^6
sbit SEG7  =P0^7
sbit SEG8  =P1^0
sbit SEG9  =P1^1
sbit SEG10 =P1^2
sbit SEG11 =P1^3
sbit SEG12 =P1^4
sbit SEG13 =P1^5
sbit SEG14 =P1^6
sbit SEG15 =P1^7
sbit SEG16 =P2^0
sbit SEG17 =P2^1
sbit SEG18 =P2^2
sbit SEG19 =P2^3
```

```
sbit SEG20 =P2^4
sbit SEG21 =P2^5
sbit SEG22 =P2^6
sbit SEG23 =P2^7
;*******************************************************************************************

;======Interrupt==============================
       CSEG AT 0000H
       LJMP start

       CSEG AT 000BH
       LJMP int_t0

;======register==============================
lcdd_bit SEGMENT BIT
       RSEG lcdd_bit
       OutFlag:       DBIT 1          ;the output display reverse flag

lcdd_data SEGMENT DATA
      RSEG lcdd_data
      Com0Data0:     DS   1
      Com0Data1:     DS   1
      Com0Data2:     DS   1
      Com1Data0:     DS   1
      Com1Data1:     DS   1
      Com1Data2:     DS   1
      Com2Data0:     DS   1
      Com2Data1:     DS   1
      Com2Data2:     DS   1
      TimeS:         DS   1

;======Interrupt Code========================
t0_int SEGMENT CODE
     RSEG  t0_int
     USING 1
;*****************************************************************
;Time0 interrupt
;ths system crystalloid is 22.1184MHz
;the time to get the Time0 interrupr is 2.5mS
;the whole duty is 2.5mS*6=15mS, including reverse
;*****************************************************************
int_t0:
    ORL  TL0,#00H
    MOV  TH0,#0EEH
    PUSH ACC
    PUSH PSW
```

```
    MOV   PSW,#08H
    ACALL OutData
    POP   PSW
    POP   ACC
    RETI

;======SUB CODE================================
uart_sub SEGMENT CODE
        RSEG  uart_sub
        USING 0
;*******************************************************************
;initial the display RAM data
;if want to display other,then you may add other data to this RAM
;Com0:   Com0Data0,Com0Data1,Com0Data2
;Com1:   Com1Data0,Com1Data1,Com1Data2
;Com2:   Com2Data0,Com0Data1,Com0Data2
;********************************************************************
InitComData:                        ;it will display "11111111"
    MOV Com0Data0,#24H
    MOV Com0Data1,#49H
    MOV Com0Data2,#92H
    MOV Com1Data0,#92H
    MOV Com1Data1,#24H
    MOV Com1Data2,#49H
    MOV Com2Data0,#00H
    MOV Com2Data1,#00H
    MOV Com2Data2,#00H
    RET

;*********************************************************************
;reverse the display data
;*********************************************************************
RetComData:
    MOV  R0,#Com0Data0              ;get the first data address
    MOV  R7,#9
RetCom_0:
    MOV  A,@R0
    CPL  A
    MOV  @R0,A
    INC  R0
    DJNZ R7,RetCom_0
    RET
```

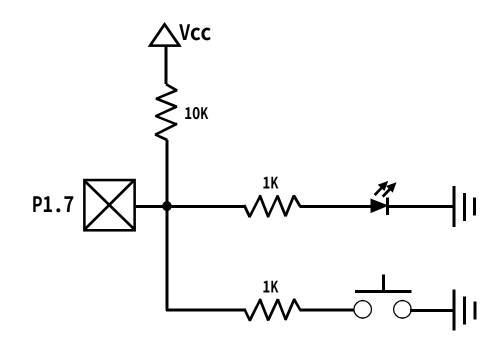
```
;***********************************************************************
;get the display Data and send to Output register
;***********************************************************************
OutData:
    INC  TimeS
    MOV  A,TimeS
    MOV  P3,#11010101B              ;clear display,all Com are 1/2VCC and invalidate
    CJNE A,#01H,OutData_1          ;judge the duty
    MOV  P0,Com0Data0
    MOV  P1,Com0Data1
    MOV  P2,Com0Data2
    JNB  OutFlag,OutData_00
    MOV  P3,#11010111B              ;Com0 is work and is VCC
    RET
OutData_00:
    MOV  P3,#11010100B              ;Com0 is work and is GND
    RET
OutData_1:
    CJNE A,#02H,OutData_2
    MOV  P0,Com1Data0
    MOV  P1,Com1Data1
    MOV  P2,Com1Data2
    JNB  OutFlag,OutData_10
    MOV  P3,#11011101B              ;Com1 is work and is VCC
    RET
OutData_10:
    MOV  P3,#11010001B              ;Com1 is work and is GND
    RET
OutData_2:
    MOV  P0,Com2Data0
    MOV  P1,Com2Data1
    MOV  P2,Com2Data2
    JNB  OutFlag,OutData_20
    MOV  P3,#11110101B              ;Com2 is work and is VCC
    SJMP OutData_21
OutData_20:
    MOV  P3,#11000101B              ;Com2 is work and is GND
OutData_21:
    MOV   TimeS,#00H
    ACALL RetComData
    CPL   OutFlag
    RET
```

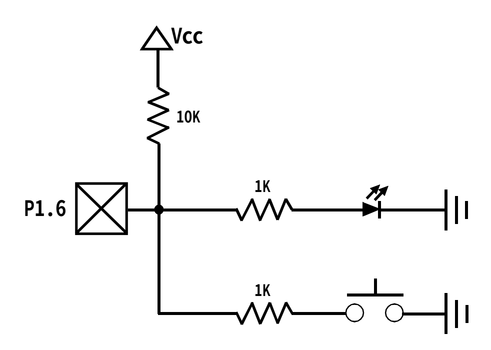
```
;======Main Code===============================
uart_main SEGMENT CODE
        RSEG  uart_main
        USING 0

start:
            MOV   SP,#40H
            CLR   OutFlag
            MOV   TimeS,#00H
            MOV   TL0,#00H
            MOV   TH0,#0EEH
            MOV   TMOD,#01H
            MOV   IE,#82H
            ACALL InitComData
            SETB  TR0
Main:
     NOP
     SJMP Main

END
```

附录D: 一个 I/O 口驱动发光二极管并扫描按键





利用STC12C5201AD系列单片机的I/O口可设置成弱上拉,强上拉(推挽)输出,仅为输入(高阻),开漏四种模式的特性,可以利用STC12C5201AD系列单片机的I/O口同时作为发光二极管驱动及按键检测用，可以大幅节省I/O口。

当驱动发光二极管时，将该I/O口设置成强推挽输出，输出高即可点亮发光二极管。

当检测按键时，将该I/O口设置成弱上拉输入，再读外部口的状态，即可检测按键。

附录E:　STC12C5201AD系列单片机应用注意事项

关于复位电路：

晶振频率在12M以下时:可以不用外部复位电路，原复位电路可以保留，也可以不用，不用时复位脚可经过1K电阻短接到地,或者直接短接到地。不过建议设计时PCB板上保留R/C复位电路，实际使用时再决定用或不用。

关于时钟：

如果使用内部R/C振荡器时钟(8MHz～16MHz,制造误差加温漂),XTAL1和XTAL2脚浮空.

如果外部时钟频率在27MHz以上时,建议采用实际基本频率就是标称频率的晶体,不要采用三泛音的晶体(基本频率是标称频率的1/3),因为外围参数搭配不当,时钟往往振荡在标称频率的1/3,即基频.或直接使用外部有源石英晶体振荡器,时钟从XTAL1脚输入,XTAL2脚必须浮空.

关于I/O口：

少数用户反映I/O口有损坏现象,后发现是

有些是I/O口由低变高读外部状态时,读不对,实际没有损坏,软件处理一下即可

是因为1T的8051单片机速度太快了,软件执行由低变高指令后立即读外部状态,此时由于实际输出还没有变高,就有可能读不对,正确的方法是在软件设置由低变高后加1到2个空操作指令延时,再读就对了.

有些实际没有损坏,加上拉电阻就OK了

是因为外围接的是SPI/I2C等漏极开漏的电路,要加10K上拉电阻.

有些是外围接的是NPN三极管,没有加上拉电阻,其实基极串多大电阻,I/O口就应该上拉多大的电阻,或者将该I/O口设置为强推挽输出.

有些确实是损坏了,原因:

发现有些是驱动LED发光二极管没有加限流电阻,建议加1K以上的限流电阻,至少也要加470欧姆以上

发现有些是做行列矩阵按键扫描电路时,实际工作时没有加限流电阻,实际工作时可能出现2个I/O口均输出为低,并且在按键按下时,短接在一起,我们知道一个CMOS电路的2个输出脚不应该直接短接在一起,按键扫描电路中,此时一个口为了读另外一个口的状态,必须先置高才能读另外一个口的状态,而8051单片机的弱上拉口在由0变为1时,会有2个时钟的强推挽高输出电流,输出到另外一个输出为低的I/O口,就有可能造成I/O口损坏.建议在其中的一侧加1K限流电阻,或者在软件处理上,不要出现按键两端的I/O口同时为低.

关于电源：

在电源两端应该加一个47uF以上的电解电容和一个0.1uF的小电容,进行电源去耦滤波.

STC12C5205/5206,STC12LE5205/5206下载用户程序时需将P1.0/P1.1短接到地

附录 F: 如何采购

请尽量从宏晶深圳直接采购，以确保质量和服务，零售 1 片起，您从银行汇款，或网上汇款，我方安排快递发货，正常 1-3 天可以收到。

TEL：0755-82948411,82948412

FAX：0755-82944243